CORE JAVA

NOTES

BY

KV. RAO

SATHYA TECHNOLOGIES

SRI RAGHAVENDRA XEROX

Software Languages Material Available

Beside Bangalore Ayyangar Bakery, Opp. C DAC, Ameerpet, Hyderabad.

Cell: 9951596199

K.V.R

Java

Class-1.

160/-

Definition of S/W :-

- A software is a collection of programs meant for performing some operations.
- In real world we have many no. of operations and these are classified by their industry experts in three types.
- classifying of operations are nothing but classfying the S/W and they are of three types -
  - (a.) system software
  - (b.) Appl^n software.
  - (c.) Internet software.

(a.) system software :-

In real world the following of the three projects comes under system software —

(i) Development of device drivers [soft. for printer, scanner, etc].

(ii) Development of language compilers, interpreters etc.

(iii.) Development of Real time operating system (RTOS) [Windows OS, Linux, Unix, Solaries, etc]

- To develop these system software projects we use the following language -
  - C
  - 8086 (Microprocessor)
- These language are popular for development of system software projects because they contains lots of features which are used to intract with Hardware devices directly.

(b.) Application software :-

The purpose of appl^n software is to develop software or project for organization.

Some of organizations are —

- Financial sector (Banking, Insuarance, etc)
- Healthcare ,, (Hospitality, Diagonostics etc)
- Transpotation ,, (Airlines, Railways, etc)
- Educational ,, (college, school, University, etc)

. . . . .

. . . . . .

. . . . . .

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

In real industry to develop Appl^n Software for an organization we use two types of Technology —

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

1. front end Technology
2. Back-end Technology

1. front end Technology :-

- The purpose of front end technology is to develop GUI application. (Look & Feel based appl^n)

- We have two types of front end technology -

(a) Platform dependent front end Technology

ex- Developer-2000 (oracle corp.)
Visual Basic (Microsoft)

- If we develop any GUI Appl^n by using about of two technology then such GUI appl^n runs on Microsoft provided O.S. only ; but not able to run on Non-Microsoft O.S. (Unix, Linux, Solaries, etc)

(b) Platform idependent front end Technology

Ex - AWT (abstract windowing Tool Kit)
Applet
Swings

} SUN-MS — Java Concept

- If we develop any GUI appl^n by using above concept then such appl^n runs on each & every OS.

2. Back-end Technology :-

The purpose of Back-end Technology is to get the data Persistency (storing the data Permanently).

- In real time we can achieve the data Persistency by using two approaches they are -
  a.) By using files
  b.) By using Database s/w

- If we store data Permanently in the form of Files; then the data of file May be Manupulated by any un-authorise user because files concept of any Language doesnot contain security in the form of User-names & Passwords. Hence industry is not recomended to store data in form of file; and Highly recommended to store the data in form of database s/w; because Most

- most of data-base s/w provides security in form of user-names & passwords.

(c) Internet software :-

The purpose of internet software is to develop distributed application (runs in the context of browser/WWW and whose results are sharable across the Universe)

To develop distributed application we have two types of Languages/Technologies they are –

a.) Java – Developed at SUN MICROSYSTEM, USA (handed over Oracle corporation.)

b.) .Net – Developed at Microsoft, USA.

Hence Java is one of the internet s/w developed at SUN Microsystem, release to the real world and used by all programmers for development of every application & More popular for development of distributed application.

History of Java

Class-2

- Java is one of the programming Language over technology treated by all the programmers as internet s/w & it is used to developed each & every Appl^n and More popular for development of distributed appl^n.
- The Java s/w developed at SUN Microsystem INC, USA under the guidance of James Gosling ("Father of Java")
- The Java s/w started its development in the early of the year 1990 & released in the Middle of the year 1991. On the Name of OAK.
- And it has taken 18 Months ~~of~~ to develop. (Scientifically it is the one of the tree name)
- In the year 1990 & 1991 the OAK was popular for development of device drivers related s/w (For Medanical electronic, automobile devices) but not popular for developing appl^n s/w & internet s/w.
- This is one of the industry experts openion.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- Based on experts openion Sun Microsystem experts OAK s/w & release to real industries in year 1995 on the name of Java. (scientifically one of the coffee seed name)

- In the real industry the Java s/w is available in three category. They are —
  J2SE (Java 2 Standard Edition)
  J2EE (Java 2 Enterprise Edition)
  J2ME (Java 2 Micro/Mobile Edition)

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- In the industry J2SE used for development of client side application, J2EE used for development of server side application & J2ME used for development of wireless/Micro/Mobile application.

Versions of Java

Versions Table :-

Java Technical version	Java Language version
JAVA	JDK 1.0 JDK 1.1
JAVA 2	JDK 1.2 JDK 1.3
	JDK 1.4
JAVA 5	JDK 1.5 (Tiger s/w)
JAVA 6	JDK 1.6
JAVA 7	JDK 1.7
JAVA 8	JDK 1.8

under SUN Micro system

under Oracle Corporation

- At present MNC's are using JDK 1.5/1.6 and we download JDK 1.5/1.6/1.7/1.8 from
  www.sun.com
  www.oracle.com

Features of Java

class-3

Features of a Language or nothing but the services or facilities provided by the Language developers in the Language which are used by programmers for development of real time applications

- In Java programming we have 13 features. They are —

1. Simple
2. Platform independent
3. Architectural Neutral
4. Portable
5. Multithreaded
6. Networked
7. Distributed
8. High Performance
9. Highly interpreted
10. Robust (strong)
11. Secured
12. Dynamic
13. OOPL

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

1. Simple :-

Java is one of the simple programming Language because of the following four technical factors —

(a) Java Programming eliminates a complex concept called pointers so that we get less application development time and less execution time because of the magic of byte code.

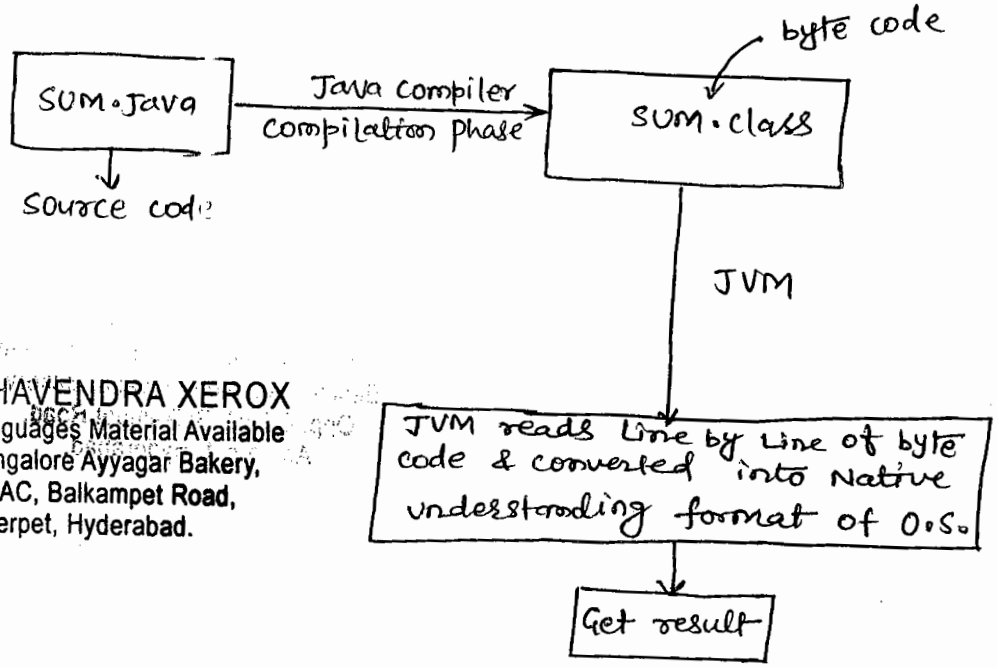
* Definition of Byte code —

A Byte code is the set of optimized instruction generated by Java compiler during compilation phase and the nature of byte code is more powerful than pointer concept.

* Definition of JVM :-

JVM is one of the program developes by Sun Microsystem Programmers available as a part of Java s/w and whose role is reading the line by line byte code & converted into native understanding format of O.S.

· Consider the following diagram —

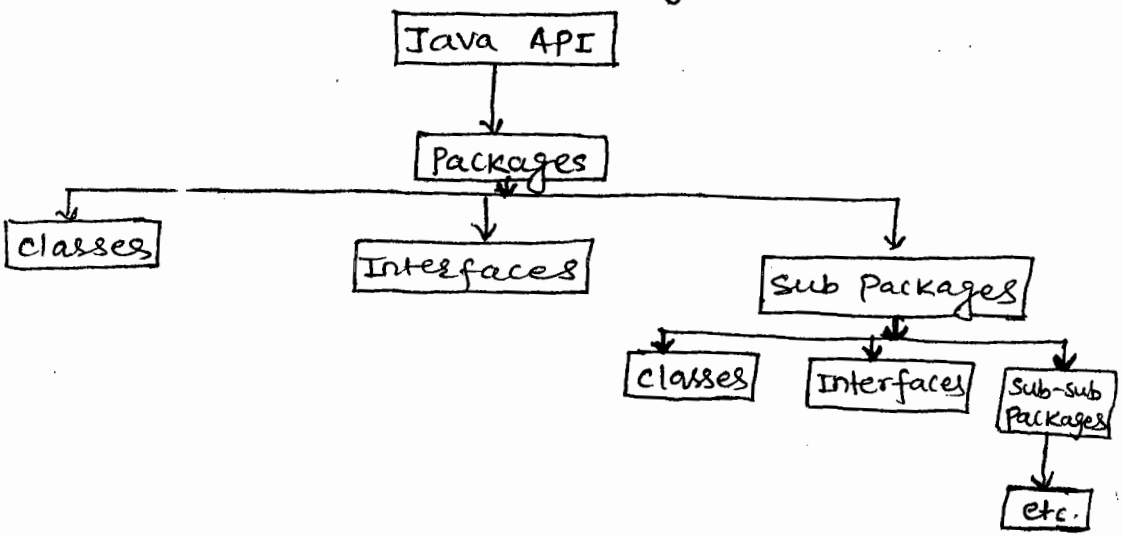


SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

(b) · Java programming provides richest set of API (Appl^n prog. interface).

defination of API :-

- An API is a collection of Packages.
- A package is collection of classes, Interfaces & sub-package. A sub packages interms contains collection of classes, interfaces & sub-sub packages etc.



(c) Java programming provides ~~building~~ built-in facility called garbage collector which will collect unused memory space to improve the performance of Java based application.

Defination of garbage collector :-
A G.C. is one of the system background Java program which are running along with our regular Java prog. for collecting un-used memory space to improve the performance of Java-based appl^n.

(d) Java programming provides uses friendly syntaxes. so that we can write error free program within the limited span of time.

class-4

(2) Platform Independent :-
A Language o technology is called to be P.I. if & only if ~~whose appl^n~~ whose appl^n is runs on E & E O.S.

Properties :-

(1) A Language / Technology related Datatype must take same amount of memory space.

(2) A Language / Technology must contain some special internal programs whose role is converting native understanding format of one O.S. into another native understanding format of another O.S.

- An another Program never satisfy those Properties then called platform dependent and Java is satisfy only those properties therefore Java called Platform independent.

Note :- Since Java based application are running on any type of O.S. sun microsystem developer released a slogen called WORA (write once run anywhere).

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

<3.> Architectural Neutral (A.N):-

Definition:-

A Lang/Technology and whose application are said to be Architectural Neutral if and only if they runs on Each & every Processors without considering their Architecture.

Note:-

In order to say a language Architectural Neutral; it has to satisfy following properties:-

(a.) A Lang/Technology must contain some special Program and whose role is converting the key of on Processor into the key of another processor.

(b.) No. Java Programming Language will not satisfy the above property and hence they are called Architectural dependency.

(c.) Java Language is treated as A.N. b/c Java Language satisfy above property (1)

Note:-

The Basic aim of SUN MICROSYSTEM developers is that "Java Programmer need not to think about types of O.S. & Type of Processor" being used and Java satisfy; A sologen called WORA.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

<4.> Portable -

- A Portable Appl^n runs on every O.S. & every Processor.
- Portability = Platform independent + Architectural Neutral
- Non-Java program in Language & whose appl^n is treated as non-portable; because they are unable to run where O.S. is different, and processor is different.
- Java language and whose application are portable because Java application runs on every O.S. & every processor.

class-5

<5.> Multi-threading :-

- The AIM of Multi-threading concept is to provide Concurrent execution (Parallel/Simultaneous execution)
- A flow of control is known as thread.
- The purpose of thread is to execute the Logic of Java program which is written in the form of user define Method concurrently.
- If Java Programs contains multiple threads then it is called multithread.
- The Languages like C/C++/Pascal/Oval etc. or treated as single threaded modelling Languages because there execution environment contains single flow of control sequential execution
  For development of thread based appl^n.
- The Language Like Java is treated as multithreaded modelling Language whose execution environment contains multiple flow of control provides concurrent execution, less execution time & containing an effect of API. for development of thread based applications.
- Whenever we developed any Java program there exist two types of thread - (a) Fore ground thread (FGT)
  (b) Back ground thread (BGT)

FGT - Always executes Logic of the Java Program written in the form of user define Methode concurrently.

BGT - Always monitors the execution status of FGT.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- The real time implementation of Multithreading is that to developed real world server software.

- In other words All the real world server software developed by third party server vendors in Java Language with Multithreading concept.

- Multithreading of Java is one of the specilize form of Multitasking of OS.

Q. How do you Justify -
"Each & every Java Program is Multi-threaded"?
When we execute the Java Program, Logic of Java Program is executed by one of the thread known as FGT.
Internally to monitor the execution status of fore ground thread one more thread is created known as BGT.
So, A Java Program is by default contains two thread
Hence; every Java Program is Multithreaded.

<5> "High-Performance" :-

Java is one of the high performance Language because of two important reasons—

1. Magic of byte code :-
2. Magic of garbage collector - which collects unused memory space & improve the performance of Java based application.

(7) "Robust" :- (strong)

- Java is one of the strong programming Language because of inbuilt facility called exception handling.

Definition of exception -
Runtime error of Java Program are known as exceptions.
Exception always generates system error messages which are understandable by Programmer but not by appl^n user.
Industry is highly recommended to generate user friendly error message, by making use of exception handling facility.

Definition of exception Handling-
The Process of converting system error Messages into user friendly error message is known as exception handling.

CTE	C. phase	SEM
LE	E. phase	M.I/
RTE	I. phase	I/I.P

The Languages like C/C++/oval/etc and whose appl^n are not robust because these Languages does not contain exception handling facility.

Class-6

(8.) "Highly Interpreted":-

- In the initial versions of Java compilation phase is one of the Fastest phase & interpretation phase is one of the slowest phase. This is the testing programmer openion.
- Sun Microsystem developers developed a new program JIT & added as a for of JVM & whose role is to speed up the intreprotation phase by reading the total byte code at once & converted into native understanding format of OS.
- In the present versions of Java interpretation phase is more faster due to JIT then compilation phase.
- Since interpretation phase is more faster than compilation phase, Sun Microsystem developers populated Java is one of the highly interpreted programming Language.

(9.) "Secured":-

Security is one of the Principle used by all the Lang. Programmer to protect the confidential info from unauthorised users.

- In the case of Non-Java Projects; Non Java Prog. devlopers needs to write there own code; for providing the security and it is complex process; because Non Java Prog. Lang. does not contain any Library for providing the security and Hence these Lang. are called unsecured.

- In the case of Java Programming; For devloping Java Project and ~~[illegible]~~ provide the security of Java Prog. need not to with their own code b/c the Java programmer need not to own code because Java Programming provide it dedicated API for Providing Security.

- Since; Java Programming contains dedicated API for Security Java language is treated as Secured Programming Language.

(10.) "Dynamic":-

- In most of the Programming Language we have 2-types of memory allocation techniques. They are –
  - (a.) static/compile-time Memory allocation
  - (b) Runtime/Dynamic Memory allocation
- In static memory allocation memory space is created for the i/p of program during compile time.
- Due to This memory allocation technique we get two Limitation:–
  1. waste of memory space.
  2. Loss of Data.

  To overcome these limitation we use dynamic memory allocation technique.
- If Dynamic memory allocation technique we allocate the memory space during execution time/ run-time.
- Java Prog. follows only dynamic memory allocation and never follows static memory allocation, hence sun micro system develops dynamic Programming Languages.
- In Java Programming to allocate the memory space dynamically we use an operator called New and its called dynamic memory allocator.

(11.) "Networked":-

The AIM of Networking is to share the data betn multiple Machine which are located either in same or it different Network.

- As a Part of Java Programming we have two types of N/w appln -- (i) intranet Appln
  (ii) internet Appln

- Intranet Appl^n can be accessed with-in the Limited distance and these Appl^n are preferred small scale organization.
- Programmatically intranet appl^n can be developed by using dedicated API present in "J2SE".

- Internet Appl^n can be accessed across the universe & there are preffered by large scale organization and they are can be developed by using dedicated API present in "J2EE"

Conclusion—
"Hence; Java programming contains seperate API for developing N/W based Appl^n and it is populated as Networked Prog. Language."

12.) "Distributed" :—

- Distributed means Appl^n results or accessible anywhere across the universe.
- In real time the Java Projects can be classified into two ways — they are—
  (i) centralized appl^n—
  (ii) De-centralized appl^n

(1) centralized appl^n—
- which appl^n runs in the content of finite no. of servers.
- operated by authorized employee.
- contains private URL.
- Perffrable by commerce organization. (Banking, insurance, etc)

(2) De-centralized Appl^n
- infinite no. of servers.
- Preferable by Non-commerce organization (yahoo, google, etc)
- accessed by both authorized or ~~non~~ un-authorized.
- Contains public URL

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

DATA TYPES :-

class-7

The Purpose of data-types is to allocate sufficient amount of memory space for the input of Program in the main-memory of computer either by following static memory allocation or dynamic memory allocation.

In every Prog. lang. data types are classified into 3-types—
(1.) Fundamental / Pre-defined / Built-In / core data type.
(2.) Derived data types.
(3.) Programmer / User / secondary / Ref. defined data types.

1. Fundamental data types —

F.D.T. developed by Language developers : exists as a Part of language & whose variable allows us to store single value but never allows us to store multiple value of same data type.

Q. - Int a;

a = 10; // valid → single value

a = 10, 20, 30, 40; // invalid – M.V.S.T.

- If we ~~choose~~ store multiple values of same types with fundamental data types then programmatically we get 2- limitation —

(a) choosing the variable is complex
(b) Remembering the " is "

To overcome these limitation we use another data types called derived data types.

2. Derived data types -

In every Programming language the concept of Array comes under derived data types.

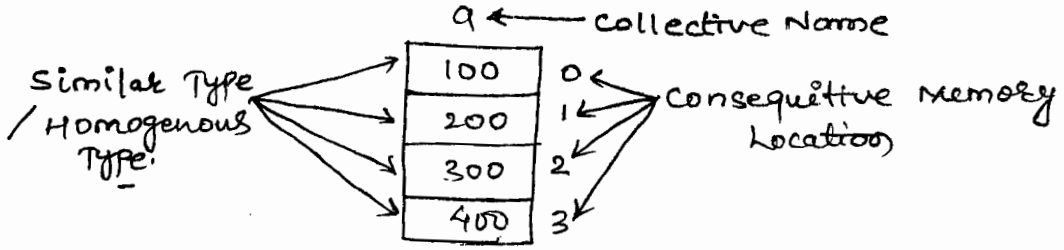
The concept of Array & whose variable allows us to store Multiple values of same type in a single variable but never allows us to store Multiple values on different types.

int a[ ] = {10, 20, 30, 40} ; // valid

int b[ ] = {10, "satya", 99.9, "A"} // invalid
because these values are different type.

a	
10	0
20	1
30	2
40	3

definition of Array



An array is collective name given to a group of consequitive memory location which are all referred by similar type / Homogenous value.

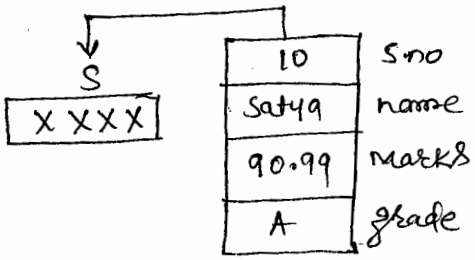
3. **Programmer/user defined data-type** —
Programmer defined data type developed by Language Programmer by using Language features & whose variable allows us to store multiple values either of same type or different type or both types.

For example - To developed programmer defined data-type in C-Language we use features Like structure (struct), Union (union), enumeration (enum) etc.

- similarly in Java programming to developed Programmer defined data type we use the features Like classes (class), interfaces (interface), enumeration (enum).

ex:-

Student S = new student ();



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Fundamental datatype in Java :-

Chs-8

In Java Programming we have 8-fundamental data type, which are categorise into 4-category.

(1.) Integer category data type.
(2.) Float ,, ,,
(3.) Character ,, ,,
(4.) boolean ,, ,,

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

1. Integer category data type :-

The purpose of integer category data types is that to allocate sufficient amount of memory space for integer data in the Main memory of computer.

integer category contains four sub-data types and they are given in following table —

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

- Here, integer category contains 4-sub datatypes and it is more important to Java programmer to choose appropiriate data type; to store data properly by calculating its range. In genral if any particular category data type contains multiple sub-data type, we need to choose suitable data type by calculating Rang.

Range of any datatype = [The No. of bits available in the Language, which is understandable by computer] / [No. of bits occupied by a Particular datatype]

Q Range of Short DT = $(2)^{16}$

$\Rightarrow 1$ to 65536

$\Rightarrow 0$ to 65535

$\Rightarrow 0$ to $\frac{65535}{2}$

$\Rightarrow 32767.5 \Leftrightarrow 32767.5$

⇒ 32767.5 ⟺ 32767.5
−0.5 +0.5
+32767 −32768

[1-bit extra because to store sign bit]

2. Float category data type :- [IEEE 754]

- The purpose of float category data type is to store real constant values in the Main memory of computer by allocating sufficient amt of memory space.
- Float category data type contains two sub-data types. which are given in the following table :-

S.NO.	datatype	size (in Bytes)	Range	NO. of decimal Places
1	float	4	+x to −(x+1)	7
2.	double	8	+y to −(y+1)	15

- If we store any real constant value in a variable of float data type, then the float data type stores the value in MM in such a way that after the dot (.) it takes (7 decimal Binary places) & real constant value must be followed by a letter f. otherwise be get compile time error.

ex- float x = 3.4f;

x
011.0110011

convert 3.4 into Binary form by using IEEE 754

2 | 3
2 | 1 — 1
0 — 1

0.4
*2
0.8 — 0
*2
1.6
*2 — 1
1.2
*2 — 1
.4
*2
8 — 0

- If we store any real const. value in variable of double data type then the double data type stores the real constant in such a way that after the dot (.) it takes 15 decimal places (Binary Bits) and real constant value may or may not be followed by a letter called 'd'.

ex - double d_1 = 3.4d;

(or)

double d_1 = 3.4;

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

d_1

011.0110 0110 0110 011

15 Binary Bits

Note :-
- If we use any real const. value directly in our Java program then it is by default treated as highest data type in float category i.e. double.

Note :- All the floating point calculation internally followed on every programming language by IEEE754 based paper calculation.

3. Character category data type :-

A character is an identifier inclosed within single quotes (' ')

ex -

'A' 'a' '$' '#' etc. ✓

'Java'
'satya Technology' } invalid

Definition of string :-

collection of character inclosed within double quotes

"Java"
"satya Technology" } valid
"A"

conclusion :- every character is a string but reverse is not true.

- To store character data in main memory of comp. in Java, C/CPP etc language we used data type called "char"

- In Java programming char takes 2 bytes because Java follows

Java programming is available in 18 international language. where as C/C++ programming language takes 1 bytes because these language follows only one international language called english and all the alphabets of English are available in ASCII character set.

Q. Define ASCII & Unicode ?

- ASCII is character set followed by those programming languages which are available in only one international languages called English & no. of character set available in ASCII are 256 (2^8).
- ASCII ex- C, CPP, pascal, COBOL follows ASCII.

Unicode character set followed by those programming languages which are available in more than one international language; and no. of characters in unicode are 65536 (2^{16}) characters.

ex- Java, .NET

4. <u>Boolean category data Type</u> :-

The purpose of this data type is to store <u>logical values</u> (feeling that existing but in reality they will not take any amt. of MM space)

ex- True False

To store logical values we use <u>boolean</u> this datatype takes 0 bytes of MM space because boolean data type of Java implemented by Sun Microsystem developers by using a genral purpose register called flip-flop and whose role is to store <u>1-bit</u> of information. (1 for true and 0 for false).

<u>Note 1</u>: All the keywords in Java must be written in small letter.

<u>Note-2</u>:- (i) The default value of int. category variables is 0 (zero).
(ii) The default value of float category variable is 0.0.
(iii) " " " character " " nothing ("\000U").
(iv) " " " boolean " " false.

Importance of variable :-

We know that with the help of data types we can allocate sufficient amount of memory space in the main memory of computer and we can store the value also. To process the values present in the memory space, it is mandatory to the programmer to give distinct names to the memory spaces. These distinct names makes us to identify values present in the spaces and they are called identi-fier.

- Identifier values can be changed from one point of time to another point of time and hence they are called variables.

defination of variable :-

- A variable is an identifier whose values are changing during execution of program.
- Hence each & every input of program must be available in main memory in the form of variables i.e. without variables one cannot represent there input.

Rules of variable :- (using variable)

out of many rules of variables. The following are the suggested rules to be used as variables —

1. Each & every first letter of variable is alphabet.
2. The length of variable should not exceed more than 32 characters.
3. No keywords is used as variable Name.
4. No special symbol to be used as variable Names except Under score (_).

Variable declaration :-

The process of allocating sufficient amount of memory space by giving unique names to the created memory space is called variable declaration.

Before using the variable in the program it must be declared otherwise we get compile time error.

Syntax –

$$\boxed{\text{datatype } V_1\ V_2\ V_3 \cdots\cdots V_n\ ;}$$

Here datatype represent either fundamental datatype or derived datatype or programmer defined data type.

- $V_1\ V_2\ V_3 \cdots\cdots V_n$ represents Java valid variable names; treated as name of the variable.

- Java programming always follows C/CPP programming declaration mechanism.

ex –

C-program declaration

```
int a, b, c;
a = 10;
b = 20;
c = a+b;
```

a	b	c
10	20	30

C++ program declaration

```
int a;
a = 10;
int b;
b = 20;
int c;
c = a+b;
```

a	b	c
10	20	30

followed by Java Program.

Variable initialization :-

The process of placing our own value, without placing default value and the memory space is created is called variable initialization.

The purpose of initialization concept is that to place common values.

Syntax :-

$$\boxed{\text{datatype } V_1 = val_1 \quad V_2 = val_2 \quad \cdots\cdots \quad V_n = val_n}$$

ex :-

```
float PI = 3.1417f;
int a=10;
boolean b=true;
```

PI	a	b
3.1417f	10	true

Constants in Java.

Definition:-

- A constant is a variable whose value can't be changed during execution of Program.
- Conceptually every common value is one of the constant value and it can be used by all the programmes but whose value can't be changed during execution of program. In Java programming to make anything as constant we use a keyword called final.
- final is a keyword which is playing an important role in three places they are -
  - a.) At variable level.
  - b.) At Method level.
  - c.) At class level.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

a.) Final at variable level -

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

At variable level we can apply final keyword in two ways -

1.) At declaration level -

Syntax -

```
final datatype v1, v2, v3 ------ vn;
```

Once the declared variable final then it permits first time assignments only and never permits second and subsequent assignment & also never permits the single modification.

ex:-

```
final float PI ; // declared variable
PI = PI + 1 ; // Invid
PI = 3.1417f + 0.0 // invalid
PI = 3.1417f ; // valid
PI = 3.141718 // invalid
```

2.) At initialization level :-

Syntax :-

final datatype v_1 = val1, v_2 = val2 . . . . v_n = valn

once we make initializer variable as final then it never permits modifications & reassignments.

ex:- final float PI = 3.1417f ;
PI = PI+1 ; // invalid
PI = 3.14178f ; // invalid.

Hence, final variable values can't be modified.

(b.) final at method level :-

In most of the programming languages to perform any operation we used the concept of function (). In Java programming function concept is known as methods.

In S/W development if we develop any method which is common for most of the Java programmers then such methods are called constant methods.

- To make the method as constant we use a keyword called final.
- Constant methods can be used by every programmer but they cannot change its definition.

Hence, once the method is final, we can't re-define/over-ridden or final method can't be override.

(c.) final at class level :-

If we don't want to give features of one class to another class then make the class definition as constant by using final keyword.

once the class is final it never participates in inheritance or final classes are not re-usable.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Note.

1. final variable value can't be modified.
2. final methods can't be overriden.
3. final classes can't be reusable.

Comments in Java programming Language

Comments are always used by improving readability / understandability / clearity of program.

- In Java programming we have 3-types of comments they are —

(1.) Single line comments

Syntax :-

```
// commenting line
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

(2.) Multiline comments

Syntax :-

```
/*
    Multiple line comment
*/
```

(3.) Java document comments -

To generate the Java doc. comments to use a tool called Javadoc.

Syntax -

```
/**
    Java doc. comments
*/
```

Development of User / Programmer define Method

We know that Each & every programming Language to Perform any type of operation we use the concept of function.

In Java Programming functions concepts is known as Methods.

To Perform any operation using Methods we have three phases. They are —

(a) A Method Must take input.

(b) A method must process the input.

(c) A Method must give the result/display the result.

The slogon of function/methods is "Write once/define once and call from anywhere"

✡ Parts of methods :-

Methods contains three Parts —

1. Method call
2. Method definition.
3. method declaration / Prototype.

— No. of approaches to define the Methods

In every Programming Language as a Part of any Program we can define a method in four ways —

1. A Method will take the input and a method will return the output.
2. A Method will not take the input and Method will not return the output.
3. A method will take the input and method will not return the output.
4. A Method will not take the input and Method will return the output.

— Syntax for defining a methods in Java :-

```
Returntype  methodName ( List of formal Parameters if any )
{
    Block of statement (S) — Business logic
}
```

Method head

Method body.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Explanation :-

Returntype - represent type of value being returned by the method. (int, float, char, double, boolean, etc) if method is not returning any value then the returntype of the method is void. (void is return-type but not a datatype).

Method Name - It represents a variable name treated as name of the method. If method contains either one word or more than one word we must write/used first word first letter is small and second and subsequent words first letter must be capital. (Known as Hangarian notation/camel case notation).

List of formal parameter :- It represents variable names which are used for holding the values which are coming from method calls. ~~Lots of statement represents~~

Bolock of statement :-

It represents the set of executable statement to be written by the programmer to perform the operation this type of logic is called business logic.

Q. Define a method in Java for computing sum of two numbers ?

The above method can be defined in four approaches they are —

Approach 1: A method will take values/inputs
&
A method will return values/inputs

Method def.	Method call

int sum (int x, int y)
{
int z = x+y;
return (z);
}
	int result1 = sum (10,20);
	int result2 = sum (5,6);
	int result3 = sum (-6, -4);

Approach: 2 - A method will not take values/inputs.
&
A method will not return values/outputs

method def.	method call

void sum ()
{
int a=10;
int b=20; // input
int c = a+b; // Parameters
Print a;b,c; // output
}
	sum ();
	sum ();
	sum ();

Approach: 3 - A method will take values / input
&
A method will not return value/outputs.

Method def	Method call

void sum (int x, int y)
{
int z = x+y;
Print x, y, z;
}
	sum (5,6);
	sum (8,9);
	sum (6,4);

Approach: 4 - A method will not take take values/ input
&
A method will return values / output.

Method def	Method calls

int sum()
{
int a=10;
int b=20; } I/P
int c = a+b; // Process
return (c);
}
	int result sum ();
	int result sum ();
	int result sum ( );

The following table gives no. of approaches to define Method V.S no. of phases in Methods —

No. of approach to define Method / phases in Method	Taking value — Method — Return value	Not taking value — Method — Not return value	Taking value — Method — Not return value	Not taking value — Method — return value
Input	outside	Inside	outside	inside
Processing	Inside	Inside	Inside	Inside
output	outside	Inside	Inside	outside

* Define a method in Java for swapping of two no's The above function should return two values after swapping but the default functionality of method will return only one values.

With the objects of Java programming one can written any no. of values.

It is advised to programmer to follow to approach 2 & approach 3.

<u>Approach 2</u> :. A method will not take value &
A method will not return values.

Method definition	Method call

void swap ()
{
int a = 10;
int b = 20; } input
int t = a;
a = b;
b = t
or
a = a+b;
b = a−b
t = a−b } process
Point a, b;
}
	swap ()
	swap
	swap ();

Approach -3 A method will take value and method will return value.

Method definition	Method call

void swap (int a, int b)
int t = a;
a = b;
b = t;
print a, b;
	swap (10, 20);
	swap (5, 6);
	swap (-10, -20);

Method overloading :-

A method is said to be overloaded if and only if method name is same & signature is different.

signature represents the following point —

(a) No. of parameter
(b) Type of parameter
(c) order of parameter

} at least one thing must be different.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

ex-

```
sum (10, 20, 30);  ———————— (1)
sum (10, 20);  ———————— (2)
float res = sum (1.5f, 2.5f);  ———————— (3)
sum (10, 1.5f);  ———————— (4)
sum (1.5f, 20);  ———————— (5)
```

Here sum(....) is called overloaded method.

Method overriding:-

Method overriding = Method heading is same & method body is different.

or;

The process of defining the original method for performing many no. of operations is called Method overriding.

ex:-

At one point of time

```
void op(int x, int y)
{
  int z = x+y;  ← original method.
}
```

At another point of time

```
void op (int x, int y)
{
  int k = x+y;  ← Redefine method.
}
```

- A overloaded method may or may not be final. Once the method is final which is not possible to override.

- At time of overloading the method we will not consider return type.
  (It can be any Return type)

- At in the case of overriding we always consider Return type.

H.W//

1. define a method for computing area of a circle.
   Hint :- [Area = πr^2]
   $\pi = 3.1417f * r * r$

2. write a method for computing area of rectangle.
   [Area = $l * b$]

3. Define a method for computing simple interest.
   $$\left[S.I. = \frac{P * i * R}{100} \right]$$

4. Define a method which will calculated area & Perimeter of square.
   Square (area = $S * S$)
   (Perimeter = $4 * S$)

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

OOPS Feature/Principle

In any MNC to develop any real world project/application we need a Language/technology and they are satisfied either Procedure oriented Principles and object oriented principles

- In other words Languages are classified into two type they are procedural oriented Programming language (POPL)
  object oriented Prog. Lang. (OOPL)

1. POPL

ex- Pascal
COBOL
ALP (8086)
PERL
upto oracle 7.3
Developer-2000
visual Basic
C etc.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- If any prog. Lang. satisfied Procedural oriented Principle then it is called POPL.
- In real world Procedural oriented Prog. lang are used for developing Appl^n/Projects related to system s/w and Appl^n s/w but not able to develop distributed appl^n; which are comes under internet s/w.
- If we use POPL for developing distributed appl^n; we get the following limitations —
  1. POPL Related appl^n used platform dependency.
  2. POPL related appl^n the data is visiting bet^n client/server side Appl^n in the form of plain text, but not in form of chiper text.
  3. POPL appl^n causes security problem
     In POPL appl^n the data is visiting bet^n client & server side appl^n in the form of byte by byte and result in poor communication.
  4. In POPL appl^n the data is around function.

conclusion:- To overcome the above limitation we use a prog. Lang. which will satisfy object oriented Principle.

OOPL –

If any prog. lang. satisfy OOPS principle then it is called OOPL.

ex:– Smalltalk
- Lisp
- Object Pascal
- Object COBOL
- C++
- Java
- .Net
- Simula
- Refil
- Ruby
- from oracle 8 onwards etc

If we developed any distributed appl^n with any OOPL (specially with Java) then we get following adv–

1. OOPL related appl^n used Platform independcy.
2. In OOPL related appl^n the data is visiting bet^n client & server side appl^n in the form of cipher text/encrypted format
3. OOPL related appl^n inhances the security.
4. In OOPL related appl^n the data is visiting bet^n client & server side appl^n all at once. and results in effective communication.
5. In OOPL appl^n the data is around objects.

Hence a programming language is said to be object oriented then it has to satisfied eight OOPS principle. They are –

a.) Classes
b) Objects
c.) data encapsulation
d.) data abstraction
e.) Inheritance
f.) Poly Morphism
g.) Dynamic Binding
h.) Message Passing.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- All the above OOPS principle are common for all OOPL but whose appl^n / syntaxes are changing from one OOPL to another OOPL.

✱ CLASS :-

- The purpose of classes concept is that to develop Programmer define data type.
- The role of Programmer define datatype is allowing us to store multiple value of same type or different type or both type.
- To develop programmer defined datatype with class concept we use a keyword class.
- E & E class of Java is programmatically treated as programmer define data type.
- E & E Java Program must starts with the concept of class without classes concept there is no Java program.

Definition-1.

A class is a collection of data members & methods or the process of binding data members and associated method in a single unit is called class.

Whenever define a class there is no memory space is created, for data members & methods of class. but whose memory space will be created when we create an object w.r.t. class. Hence classes contains logical existance and objects contains physical existance.

Data members of class are also known as attribute, field or properties. whereas methods are also known as accessories or behavious.

- In OOP we have two types of methodes. They are -

a.) member method

b.) Non-member method.

a.) member method -

A member method is available within the scope of class and it can access values of data member of class.

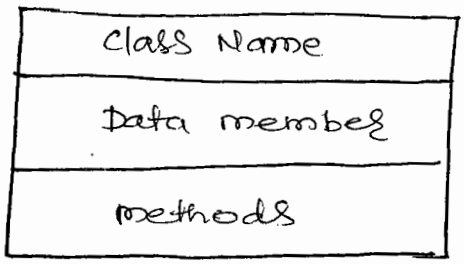
b) Non-member method -

A Non member method is one of which is not available within the class and it is not access value of data member of class.

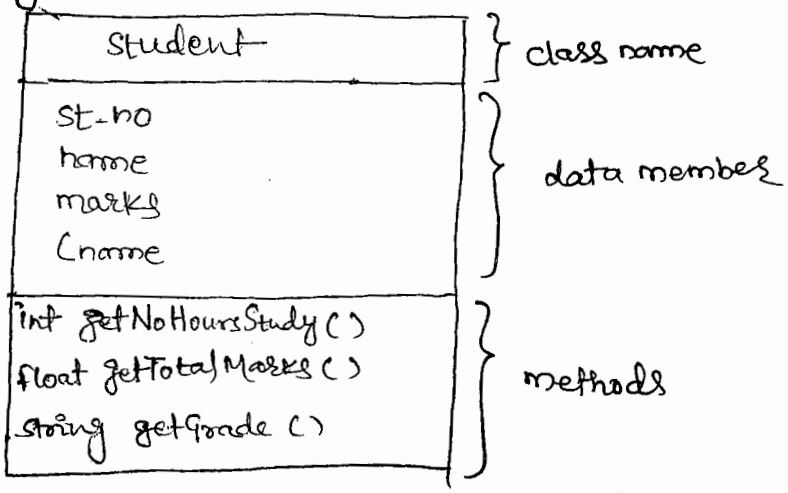
Conclusion –

- Java Prog. allows only member methods but never allows NON- member methods.

- In real world application the definition of the class will be expressed by using class diagram and whose str. is shown below



Draw a class diagram for student information



The above diagram is common for all the Programmers of OOPL but whose syntaxes will be changing from one OOPL to another OOPL.

Syntax for defining a class in Java :–

```
class <clsname>
{
    var declaration;
    method definition;
}
```

Explanation :-

- class is a keyword used for developing user define data type which will hold multiple values either of same type or different type or both type.

- <clsname> represents Java valid variable name treated as a name of a class. E & E class name acts as user / programmer defined data type. when we define a class memory space will not be created for the data members & methods but whose memory space is created when we create an object.

- variable declaration represents data members of the class. and they are always met for placing the value and they will be selected based on the class name.

- methods definition represents list of methods met for Performing some type of operation by making news of value of data members of class: and these methods will be selected based on the name of a class.

- In Java programming all the methods must be define within class difinition only. and there is no syntax is define the method outside the class.

- The definition of the class must be incorporated or Embedded within { }. and definition of class may or may not be terminated by semicolon.

write a class in Java for representing student information.

```
class Student
{
  int stno;
  String sname;
  float m1, m2, m3;
int getNoHoursStudy()
  {
    return(4);
  }
  float getTotal()
  {
  float tot = m1+m2+m3;
      return(tot);
  }
  String getGrade()
   {
    return ("Disti");
   }
}
```

Q. Define a class for Performing Sum of two numbers ?

```
class Sum
{
  int a, b, c ;

  void accept()
  {
    a = 10;
    b = 20;
  }
  void add ()
  {
    c = a+b;
  }
  void display ()
  {
   Sop ("val of a =" + a);
   Sop ("val of b =" +b);
   Sop ("val of c = "+c);
  }
}
```

Sum
a: b: c:
void accept() void add () void display ()

- A class of Java that contain many no. of methods & among them are related to taking i/p Processing ip and displaying o/p.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

2. OBJECT

- we know that when we define a class there is no memory space is created for the method & data member but whose memory space will be created when we create an object with respect to class.

- In other words in order to enter the values for the members of the class ~~first~~ be must allocate sufficient memory space for the DM of the class by creating an object.
- To do any data processing there must exist an object. Hence to create an object in Java there must exist a class definition otherwise we get compile time error.

Definition :-

1. Instance of a class is known as OBJECT.
   (Instance is nothing but allocating the suitable memory space for data members & methods).

2. E & E class variable is known as object.

3. E & E grouped item is called an object.
   (A grouped item is a variable which allows us to store multiple value either same or different or both type).

Creating an object in Java.

- Creating an obj is nothing but allocating sufficient memory space for DM & methods of class by following Dynamic Memory allocation with the help of new operator.

functions of new operator

1. It allocate suitable space for data member & methods of class.

2. It takes an add./refrence of the loaded class and place it into L.H.S. variable i.e. object name.

3. out of many ways of creating an obj in Java with new operator we can create obj. by using the following two syntaxes.

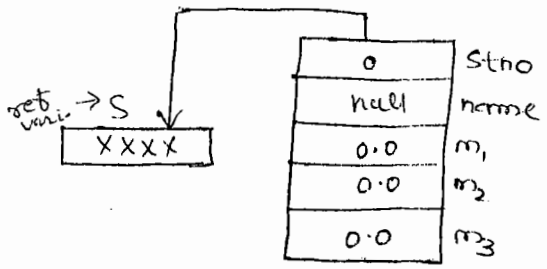
Continua...

Syntax 1:

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

ex:-

Student s = new Student ();



Syntax 2:

<clsname> <objname>; ——— (1)

<objname> = new <clsname()>; ——— (2)

Statement 1 represent object declaration whenever an obj. declared memory space is not created for DM & Method of class & the default of value declared object is NULL.

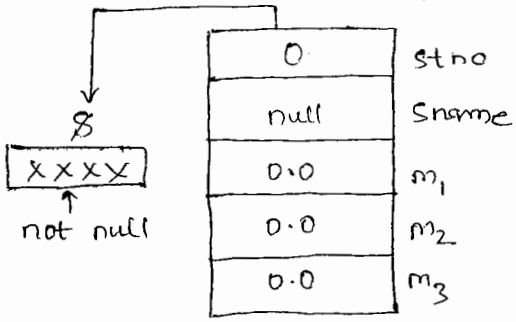
ex:- Student s; // S [null] ← Studen

Sum s_1; // S [null] ← sumclass obj. declaration

String name

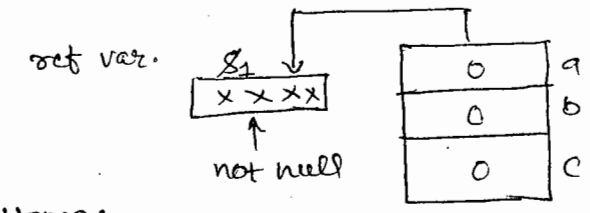
Statement 2 represents object refrencing. when an obj is refrenced memory space is created for the data member & methods of class and the default value of refrenced variable is not NULL. (refrence of loaded class).

ex- s = new Student (); // Student class obj. referencing



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

ex- S_1 = new Sum(); // Sum class obj. ref.



Hence;
- Concept of class always gives plan about our Java prog. and obj. concept gives implementation of the plan.

Q. <u>define class loader sub-system</u> (C.L.S.S.) ?

It is one of the program developed by SUN Microsystem dev. and available as a part of Java s/w and whose ~~use~~ real is that transferring or loading the dot class (.class) file from secondary memory into main memory.

Q. <u>what are the diff. betn class & obj ?</u>

Class	object
1. A class is a collⁿ of DM & Methods.	a. Instance of class is called object.
2. Classes contains logical existance.	b. Obj. contains physical existance.
3. When we created a class memory space is not created for the DM & methods.	c. when we create a object memory space is created for the DM & methods of class.
4. In a Java prog. definition of a particular class will exist only once.	d. with respect to one class defⁿ we can create multiple obj.
5. Def. of class reside in secondary memory in form of (.class) file.	e. ~~definition~~ object of particular class resides always in main memory.
6. when we execute any Java program that definition of class loaded only once in MM with the help of class loaded subsystem.	f. After loading the class defⁿ in main memory later object can be created.

Note - 1

- Objects of Java resides heap memory.
- methods of my language resides & executes in stack memory.
- figurative constant resides in associative memory.

Types of Data members in class

- In Java prog. data members of a class are classified into two types they are —
  1. Instance / non-static data members
  2. Static data members.

Instance D.m.

(1) - I.D.m. memory space created E & E when the obj. is created.

(2) - IDM always used for storing specific value which are suitable to individuals.

(3) - IDM declaration should not be preceded by static keyword.

(4) Syntax -

datatype $V_1, V_2, V_3 \ldots V_n$

(5) IDM must be accessed with respect to object name

objname . IDM name

(6) IDM values are not sharable.

(7) IDM are also known as OLDM (object level Data member)

(8) Ex - int PIN;
String name;
float marks;

Static Data member

(a) S.D.M. memory space is created only once when the class is loaded into main memory irrespective no. of obj created.

(b) - SDM are used for storing common value which are suitable for all programmers

(c) SDM declaration must be preceded by static keyword.

(d) Syntax
Static datatype $V_1, V_2 \ldots V_n$.

(e) SDM must be accessed with respect to class name (even with object also)

clsname . SDM name

(f) SDM values are sharable.

(g) SDM are also known as CLDM. (class level data member)

(h) ex —
Static String crs;
Static String capIndia;
Static float PI;

- E & E final variable value must be static but static variable value may or may not be final.

ex -

```
static final float PI = 3.1417f;

static String crs = "JAVA";
```

Types of Methods in Java

- In Java programming a class definition contain 2-types of methods. They are —
  - (a) Instance/Non-static methods
  - (b) Static methods

Instance methods

(1) I.M. are met for performing repeated operations such as reading records from file, from Database etc.

(2) I.M. can perform specific operation.

(3) programmatically ~~static~~ Instance M.D. should not be preceded by static keyword.

(4) Syntax :-

```
ReturnType methodName ( List of formal parameter if any )
{
    Block of stat. — — Business logic.
}
```

(5) E & E I.M. must be accessed w.r.t. obj. name

```
objname.ImName();
```

Static methods

(a) Static methods are met for performing one time operations such as opening the file in read mode or write mode, obtaining the database conn. etc.

(b) S.M. can perform common operation.

(c) Static method definition must be preceded by static keyword.

(d) Syntax

```
Static RetType methodName ( List of Formal parameter if any )
{
    Block of state — — Business logic
}
```

(e) E & E S.M. must be accessed w.r.t. class name (even with obj also)

```
clsname.smName();
```

IM	SM
(6) Im results are not sharable	(6)- SM results are sharable
(7)- IM are also known as Object level methods	(7) - SM are also known as Class level methods.

(8) ex-
void calTotal()
{
tot = $m_1+m_2+m_3$;
}
	(8)
	Public static void main(.....)
	{
	.....
	s1.calTotal();
	s2.calTotal();
	———
	———

Note :-

- we know that a class can contain IDM, SDM, Instance methods & static methods.
- Instance Data members & Instance methods must be accessed w.r.t. object name and static data members & static methods are must be accessed with class name (even with object also).

System.out.println(—) ——— (1)
System.out.print(—) ——— (2)

Statement(1) is used for displaying a result of Java Program on the console <u>Line by Line</u>.

Statement(2) is used for displaying the result of Java Program on the console <u>in the Same Line</u>.

Technical description :-

1. Here println and print are the two Predefine overloaded instance method present in a predefine class called print Stream.
2. To access println & print methods we need an Object of print stream class.
3. An object of Printstream class is called out is created as a static data member in an another Predefine class called System.

The following code segment gives technical description of statement (1) & (2).

```
class System
{
 ---------
static printstream out = new printStream();
}


class PrintStream
{
   void println()
   {
    ____
    ____
   }
   void print()
   {
    ____
    ____
   }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Hence;
println & print methods must be accessed as

System.out.println(__).
System.out.print(__).

1. **Ex** :- Display "Hello Java world".

```
System.out.println("Hello java world");
```

2. **Ex** :-

```
int a = 10;
System.out.println(a);   // 10
sop("val of a = " + a);  // val of a = 10;
sop(a + " is the val of a");  // 10 is the value of a
```

ex-3 - int a = 10 ; b = 20;
int c = a+b;

sop ("sum =" + c); → // sum = 30
sop (c + " is the sum"); // 30 is the sum
sop ("sum of" + a + " and " + b + " = " + c); // sum of 10 and 20 = 30

+a
↳ concatenation of a.

ex-④ Sathya Tech 'JG' Java

sop ("Sathya Tech" + ",") + ("JG") + ",") + ("Java"));

ex - '10'

sop ("c" + 10 + ",");

or

sop ("c" + "10" + ",");

Structure of Java Program

- The standard format described by language developers and used by the programmer for developing any str./Appl/Prog.

- Sun microsystem developers provides the some standards format for java programs.

```
Package info ;
Class <clsname>
{
    Data member

    method (user-defined)

public static void main (String k[])
    {
      Block of statement (s)
    }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

* Explanation of Java Program:-

1.) Package information contains coll^n of classes, interfaces & sub packages. A sub packages in terms contains coll^n of classes, interfaces & sub-subpackages etc. If we use any predefine class or interface as a part of our Java program then it is mandatory to Java Programmer to specify in which package these classes & interfaces presents otherwise we get compile time error. The default package which is imported to every Java program is Java.Lang.* known as default package.

2.) We know that E & E Java Program must starts with the concept of class with a keyword class.

3.) <clsname> presents a Java valid variable name treated as a name of class.

4.) Data member represents either instance or static & they will be selected on based on class name.

5.) User define method represents set of methods meant for performing different types of operation either once (static) or repeatedly (Instance method)

6.) E & E Java program must start executing from main(-). ~~method~~ and hence main(-) is called "Program driver".

7.) Since main(-) of Java will not return any value and hence its return type must be void (void is return type but not a data type).

8.) Since main method of Java executes only once throughout the life of entire of Java program execution and hence its nature must be static.

9.) Since main method of Java accessed/used by all the Java programmers and hence its access modifier must be Public.

10.) E & E main method of Java takes array of obj of String class.

11.) Block of statement represents set of executable statement and they inturns calls user define method either w.r.t object instance () or w.r.t. class.

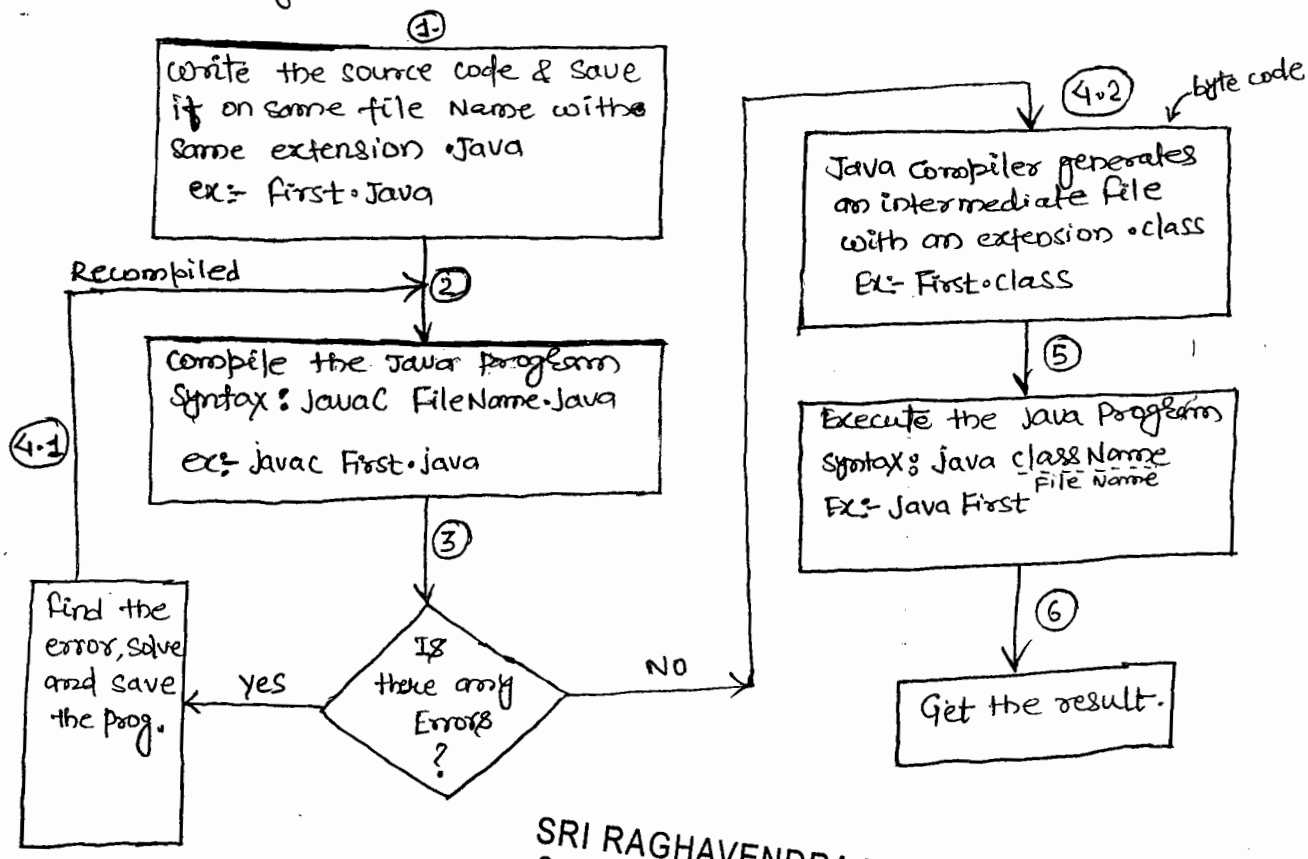
12.> The file naming Convension in Java programming is that which our class name contain main (–) it must be given as file name with an extension .Java.

Q. write a Java program which will print "Hello Java World".

```
// First.Java
Class First
{
Public static void main (String K[])
{
system.out.println ("Hello Java world");
System.out.println ("This is my first program");
}
}
```

* Steps for compiling & Executing the Java Program

The following diagram gives sequences of steps for compiling & Executing the Java Program:→



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- In the above diagram javac & java are the two tools/appl' prog. developed by SUN MICROSYSTEM developers and available as a part of JAVA-HOME\bin folder and they are used for compiling & executing the java program respectively.

Requirement Analysis for Developing Java Appl^n/Projects

- To develop a real world application we use three types of requirements they are —
  1. Software requirement
  2. O.S. requirement
  3. IDE's

(1.) S/w requirement:—

freely downloaded Java S/w (JDK 1.5/1.6/1.8) from
www.oracle.com
www.sun.com

(2.) O/S requirement:—

Any O.S. can be taken because platform independent.

(3.) IDE's requirement :- (Integrated development Env.)

- IDE is one of the third party S/w developed by third party vendors; released to the real world and used by all the programmers and the provide built-in facility such as syntax Highlighting, fixing the bugs, toggle point tracing the application.

The following are some of the IDE's —

(I.) DOS Editor, Note Pad, Notepad plus, Word Pad etc } Poor IDE's & Not used by Industry.

(II.) Editplus - ES Enterprise } Poor IDE's but Best for Learner & not used by Industry

(III.) NetBeans, Eclips, WSAD, RAD } Best IDE's used by Industry.

Setting the Path at time of compiling & executing Java Program:

Step-I -

create our own directory (E:\\9amJava\classes) and Save our program.

E:\\ 9amJava
→ classes
→ First.Java

Step-II -

Compile the Java program E:\\9amJava > First.Java ↵

We get an error

"Javac is not a recognized the command"

Because Javac tool is not found in classes folder but it is present in bin folder of Java s/w. To make use of Javac tool we need to set the path upto bin, by using Path command (It is the environmental variable name)

E:\9amJava\classes> set Path=D:\Java\JdK1.6\bin ↵

compile the program—

E:\9amJava\classes > Javac First.Java

- Ensure First.class file will be generated. In general .class file will be generated based on class definition but not generated on the basis of file name.

- Run the Java Program.

E:\9amJava\classes > Java First

→ When the above statement is executed internally the following steps will takes place —

1. Class loader sub-system Loads the specified class (eg- First.class)
2. JVM takes loaded class (eg- First.class)
3. JVM looks for main(-) because every Java prog. starts executing from main methods.
4. JVM calls main(-) w.r.t. loaded class (First) as First.main(-).

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

V.V.9

Method chaining*

The process of establishing the communication betn multiple methods perform some operations by exchanging the data/info. is called method chaining.

Rules for method chaining :-

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

(2.) one Instance () can call another Instance () directly provided both Instance () belongs to same class.

(3.) One static () can call another Instance () w.r.t. Object name without considering whether they belongs to same class or different class.

(4.) one Instance () can call another static () w.r.t. class name without considering whether they belongs to same class or different class.

Q. write a Java program which illustrate methods chaining rule.

```
// Third.java
Class Third
{
 void show()
 {
  write(); // Rule-2
  SOP("show()-Instance");
 }
 Static void disp()
 {
  SOP ("disp()-Static");
 }
 Void write()
 {
  Third.disp ( ); //Rule-4
  SOP ("write-Instance()");
 }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
psvm (String [] args)
{
  sop ("I am from main() - beg");
  Third t1 = new Third();
    t1.show(); // Rule-3
    disp(); // Rule-1  or  Third.disp();
    sop ("I am from main() - end");
  }
} // Third
```

Note :-

1. - "Internally methods chaining concepts is implemented with the helps of STACK".

2. - It is highly recommended to write a seperate Business Logic class with coll^n of Business Logic statements.
   Business Logic classes always Provides Solutions to the client requirement.

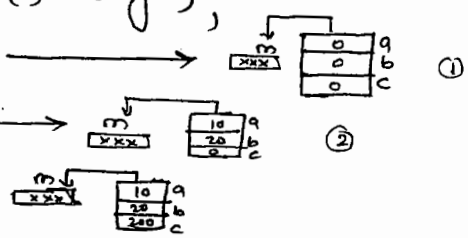
3. - It is also highly recommended to write a seperate execution Logic class with single execution Logic method (main(-)) with collection of Execution Logic statement.
   The purpose of execution Logic class is accessing the Business Logic methods.

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

```
// mulDemo.Java
class mul
{
int a, b, c;
void accept ()
{  a = 10;
   b = 20;
}
void multiply ()
{  c = a*b;
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
void disp()
{
  sop("value of a =" +a);
  sop("value of b =" +b);
  sop("value of c =" +c);
}
} mul - BLC

Class MulDemo
{
  psvm(String[] args)
  {
    sop("I am from main()-beg");
    mul m = new mul();
      m.accept();
      m.multiply();
      m.display();
    sop("I am from main().end");
  }
} // mulDemo - Execution Logic class
```



Note :-

1. - In the above program the accept() & multiply() are changing the values of objects and hence they are called <u>mutators</u>.

2. - Any type of method of Java (disp()) is obtaining the values of the object then such methods are called <u>Inspectors</u>.

- If we run the above prog. multiple times we get the same output which is not a recommended process. if we want to get different result as appl^n user we must perform the cyclic operation which is following —

1. - go to the program.
2. - change the required value.
3. - Save the Program.
4. - compile the Program.

- These steps mayn't Performed by application user, because application user is not a programmer.

- If any program satisfy the above charasteristics then such programming is called Static / hard-coded Programming. and it is not recommended in industry & highly recommended to develop dynamic programming / flexible prog.

Q. Define hard coded approach & Dynamic Approach?
- In hard coded approach input of the program mentioned within the program, Participated during compilation time & used at run-time and gives fixed output.
- In dynamic programming input to the program supplied at runtime & used at runtime & gives dynamic result.

No. of approaches to get the data dynamically

In Java Programming we have four approaches to accept the data dynamically they are —
1. Through Command prompt
2. Through Keyboard
3. Through Properties file / Resource Bundle file
4. Through XML document.

✱ length :-

length is one of the implicit variable created by Java environment and supplied to E & E Java Program for two purpose. They are —
- It finds no. of elements present in an array (size of array).
- To treat fundamental array [int[], float[], char[] etc] as Objects Internally.

Syntax

```
arrName.length
```

arrName represent name of the array variable.

① ex-

fundamental array

```
int a[] = {10, 20, 30};
SOP("No. of elements:"+a.length); // 3
for(int i=0; i<a.length; i++) //
{
  SOP(a[i]);
}
```

② ex-

```
float x[] = {1.5f, 2.1f};
SOP("No. of elements:"+x.length); // 2
for(int i=0; i<x.length; i++)
{
  SOP(x[i]);
}
```

array of obj. of string class

③ ex

```
String s[] = {"cpp", "JAVA", "oracle"};
SOP("No. of cources:"+s.length); // 3
for(int i=0; i<s.length; i++)
{
  SOP(s[i]);
}
```

1. <u>Accepting the data dynamically through Command Prompt</u> :-

The no. of values passing to the java program from command prompt are known as command line argument consider the following statement.

$\underbrace{\text{E:\textbackslash 9AM JAVA\textbackslash classes}}_{\text{Cmd prompt}}$ > $\underbrace{\text{Java mulDemo}}_{\text{Java prog}}$ $\underbrace{\text{10 20}}_{\text{values}}$

When the above statement is executed then

1. - Class Loader Sub-system Loads mulDemo into main memory along with cmd line argument (10, 20) eg.
2. - JVM takes loaded class mulDemo along with command line arguments (eg. 10, 20) and counted them (length eg. ②).

3. JVM looks for main (String K[2])

10	0
20	1

4. JVM calls the main (-) with respect to loaded class mulDemo as mulDemo.main(-).

ex2 -

E:\9AMJAVA> Java StudentInfo 10 Sathya 99.90

cmd prompt — Java Prog — values

Calling w.r.t. — Passing to

StudentInfo.main (String K[3])

Code for printing cmd line args.

```
for (int i=0; i<K.length; i++)
{
  sop (K[3])
}
```

10	0
Sathya	1
99.90	2

ex-3

E:\9AMJAVA> Java ProdInfo 10 Pendrive 88.10

cmd prompt — Java prog — values

calling w.r.t — Passing to

ProdInfo.main (String K[])

10	0
Pendrive	1
88.10	2

Code for printing cmd line args

```
for (int i=0;  i<K.length; i++)
{
  sop (K[i]);
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- Irrespective of type of cmd line arguments we enter, which are by default treated by JVM as String type.

Q. write a Java program which will accept command line arguments ? & display those cmd line arguments.

```
// CmdLine.Java
class CmdLine
{
public static void main (String K[])
 {
 System.out.println ("no. of cmd line argument:" + K.length);
 for (i=0 ; i < K.length ; i++)
 {
   system.out.println ( K.[i]);
 } // for
} // main()
} // class cmdLine
```

✱ Converting String type data into Numerical/fundamental type data

- we know that all the commandline Arguments which is entered by user taken by ~~main (~~) JVM and sending to main method & available in the form of ArrayOFObject of String class.
- on String type data we can't Perform any numerical operation & hence it is highly desirable to convert string type data into Numerical/fundamental type data.

The Following table gives fundamental Data-types in Java, corresponding wrapper class name & conversion methods from numerical string type into fundamental/Numerical Type data

Fundamental type data	Wrapper class Name	conversion method from Num. String into Fund/Num. Data Type
byte	Byte	Public static byte parseByte (string)
short	Short	public static short parseShort (string)
int	Integer	public static int parseInt (string)
long	Long	public static long parseLong (string)
float	Float	public static float parsefloat (string)
double	Double	public static double parseDouble (string)
char	Character	——— ———
boolean	Boolean	public static boolean parseBoolean (string)

- In general E & E wrapper class (except character) contains the following generalized method :—

Wrapper class Name

Public static xxx parseXxx (string)

"Here Xxx represents any F.D.T. except char".

Note-

Q. what are the three points to remember at the times of using methods in Java.

1. Decide what Type of a method we are using & present in which class (Ins/static method)
2. Decide what type of Parameter(s) a method is taking & pass its value(s).
3. Decide what type of Return-type a method is returning & hold the result in the variable of that ret. type.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

ex:-

```
String s1 = "100";
s1 = s1+1;
sop(s1); // 1001
```

*

```
int x = Integer.ParseInt(s1);
```

```
x=x+1;
sop(x); // 101
```

ex:-

```
String x = "1.5f";
float y = Float.parseFloat(x);
y=y+1;
sop(y); // 2.5
```

ex:-

```
String s[] = {"10", "20"};
sop(s[0] + s[1]); // 1020
```

s	
10	0
20	1

```
int x = Integer ParseInt(s[0]);
int y = Integer ParseInt(s[1]);

int z = x+y;
sop("sum =" +z); // sum =30
sop("sum =" +x+y); // sum = 1020
sop("sum =" +(x+y)); // sum =30
sop("sum =" +x-y); // ..... error
sop("mul =" +x*y); // 200
sop("Div =" + y/x); // 2
sop("sub =" +(x-y)); // -10
```

Q write a Java Program which will calculate multiplication of two no's by accepting dynamic value from Cmd prompt.

```
// mul1.java
class Mul
 int a, b, c
 void accept (int p, int q) // here 'p' & 'q' are called formal
 {                                                  Params
  a = p;
  b = q;
 }
 void multiply ()
  {
 c = a * b;
 }
 void disp ()
 {
  sop ("value of a =" + a);
  sop ("value of b =" + b);
  sop ("value of c =" + c);
 }
} // mul - Business logic class
 class Mul1
 {
  P.s.v.m (String [] args)
  {
   sop ("I am from main() - beg");
 // convert cmd line args into int type values
     int x1 = Integer.parseInt (args[0]);
     int x2 = Integer.parseInt (args[1]);
 // Pass the cmd line args to BLC obj
          Mul m = new Mul();
          m.accept (x1, x2);
          m.multiply ();
          m.disp ();
     sop ("I am from main() - end");
 }
} // mul1 — exec Logic class
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

compile the above program-

Javac Mul1.Java

ensure mul.class & mul1.class must be generated.

- Run the above program by Passing two command line args —

cmd line arguments → Java mul1 9 8

O/P → I am from main() -- beg
val of a = 9
val of b = 8
val of c = 72
i am from main() -- end

Q. Define a wrapperclass ? what is its purpose ?

For E & E predefine-datatype there exist a predefine class & it is known as <u>wrapper class</u>.

The purpose of wrapper classes is that to convert ~~new~~ numerical string value into numerical/fundamental type values.

Q. write a Java program which will calculate square of a given number ?

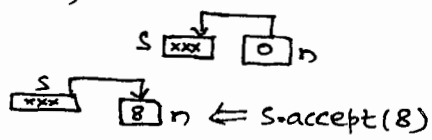
Hint:- square of n = n*n
square of (4) = 4*4
= 16
square of (-5) = 25

```
// sqDemo.Java
class Square
{
 int n;
 void accept (int x)
  {
   n = x;
  }
 int calsquare()
 {
   int res = n*n;
   return res;
 }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
void disp ()
{
  int res = calsquare();
  sop ("square ("+n+")="+res);
}
} //square - BLC
Class SqDemo
{
  P.S.V.m (String K[ ])
  {
    if ( K.length != 1)
    {
      S.O.P. ("Please enter one value:");
    }
    else
    {
      // convert cmd line args into fundamental type
      int x = Integer.parseInt (K[0]);
      Square s = new square();
        s.accept (x);
        s.disp();
    } //else
  } // main()
} // SqDemo - ELC
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.



Q. write a java Program which will interchanged two values ?

Hint:-

Logic-I	a	b	Logic-II
	10	20	
int t = a;			a = a+b;
a = b;			b = a-b;
b = t			a = a-b;
(with Temporary var.)			(without temp. var.)

```
// SwapDemo.java
class swap
{
 int a,b;
 void accept ( int a, int b)
 {
  a=p;
  b=q;
 }
 void SwapValues ()
   { int t=a;
       a=b;
       b=t;
   }
 void disp ()
   {
    sop ("val of a=" +a);
    sop ("val of b=" +b);
   }
}
class SwapDemo
 {
  p.s.v.m (String [] args)
  {
   if (args.length ! =2)
   {
    sop ("plz enter two values:");
   }
   else
    {
     int x1 = Integer.parseInt (args[0]);
     int x2 = Integer.parseInt      (args[1]);
      swap s = new swap();
        s.set (x1,x2);
```

```
s.o.p. ("————");
s.o.p. ("original values");
s.o.p. ("————");
s.disp();
s.o.p. ("————");
s.swapValues();
s.o.p ("————");
s.o.p ("swapped values");
s.o.p ("————");
s.disp();
s.o.p.("————");
}
}
}
```

Q. write a java program which will decide whether the given no. is even or odd.

```
class EvenOdd
{
int n;
void set (int x)
{
n=x;
}
void decide()
{
if (n==0)
{
System.out.println("invalid input !!!");
}
else
{ if (n%2==0)
{ sop (n+" is even no.");
}
else
{ sop (n+" is odd no");
}}}}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
class EvenOddDemo
{
p.s.v.m (string [] args)
{ if (args.length! =1)
  {
   System.out.println(" enter only one value");
  }
 else
  { int p = Integer.parseInt (args[0]);
    EvenOdd eo = new EvenOdd ();
    eo.set(p);
    eo.decide();
} } }
```

Q. write a java program which is decide whether the given no. is -ve, +ve or zero.

```
class Sud
{
  int n;
  void set (int x)
  {
    n=x;
  }
  void decide ()
  {
    if (n==0)
    {
      sop (n+" is zero");
    }
    else
      {
        if (n>0)
        {
          sop (n+" is negative value");
        }
        else
        {
          sop (n+" is positive value");
        } //else-inner
      } // else-outer
  } //decide ()
} // BLC
```

```
class PosNegDemo
{
  P.S.V.m (String [] args)
  {
    if (args.length != 1)
    {
        S.O.P. ("enter one value");
    }
    else
    {
      int P = Integer.parseInt (args [0]);
      Sud S1 = new Sud();
      S1.Set (P);
      S1.decide();
    } //else
  }// if
}// ELC
```

Another way:

```
// PNZDemo.java
class PNZ
{
  int n;
  void set (int x)
  {
    n=x;
  }
  String decide ()
  {
    if (n==0)
    {
      return ( n+" is zero");
    }
    else
    {
      if (n>0)
      {
        return (n+" is Positive no");
      }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
else
  {
    return (nt" is negative value:");
  } // else-inner
} // else-outer
} // decide()
        void disp (String s)
        {
          S.O.P (s);
        }
  } // BLC

Class PNZDemo
{
  P.S.V.M (String args[])
  {
    if (args.length != 1)
     {
      SOP ("Plz enter one value");
     }
    else
     {
      int p = Integer.parseInt (args[0]);
        PNZ PO = new PNZ();
         PO.set(P);
         String res = PO.decide();
          PO.disp(res);
      } //else
    } // if
} // Class - ELC
```

Note -

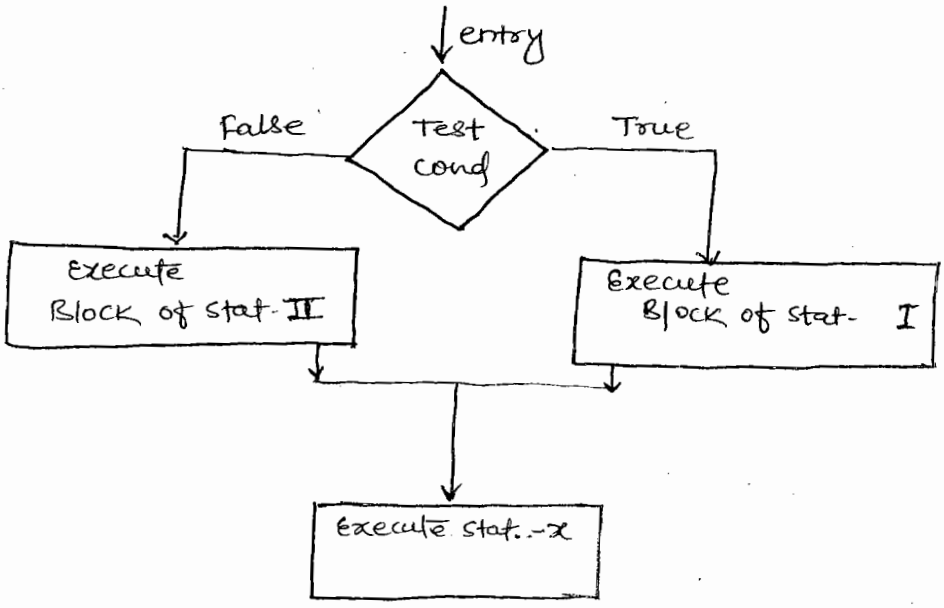
1.- A method of java is not only returning fundamental data type but it can also return User-define data types. (class-types as return type)

2.- A method of java is not only taking fundamental types but it also takes class types.

Control Structure

The purpose of control structure is that to execute either some operation or some other operation only once depends on condition evaluation (conditional statement) or Excute some other operation repeatavely for finite no: of times untill test condition becomes false. (looping statement)

Flow chart for conditional statement:-

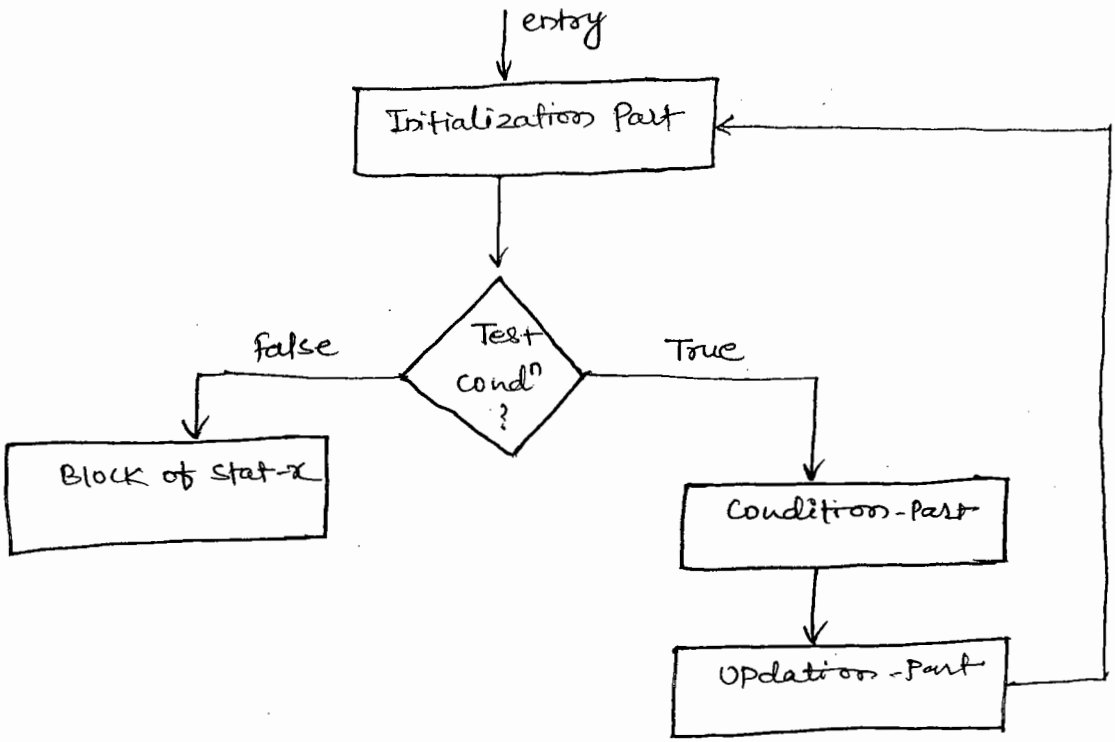


The conditional statement are -

(a) If
(b) If-else
(c) Switch

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Flow chart of Looping statement :-



The Looping statement are

(a) - while
(b) - do-while
(c) - for

Syntax - while :-

```
while (Test cond^n)
{
  Block of stat.(s)
}
                  stat. - x
```

do-while

```
do
{
  Block of stat (s)
}
while (Test-cond^n);
```

for

```
for ( initialization ; cond^n part ; updation part)
{
  Block of stat (s)
}
statement - x
```

Q. write a java program which will print 1 to n-no's accept n-value from cmd prompt and it should be Positive.

```
// IncrDemo.java
class Incr
{
 int n;
 void set(int x)
  {
   n=x;
  }
 void generate()
  {
   if (n<=0)
   {
    sop (n+" is invalid input, plz enter +ve value");
   }
   else
    {
     int i=1;
       while (i<=n)
        {
         sop (    i);
         i= i+1;
        } // while
    } //generate ()
  } // BLC
class IncrDemo
 {
  p.s.v.m (string [] args)
  {
   if (args.length != 1)
    {
     s.o.p. ("enter one value atleast");
    }
   else
    {
     int a = Integer.parseInt (args[0]);
     Incr io = new Incr();
      io.set(a);
      io.generate();
    } //else
  } //main
}//generate
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. write a Java Program which will accept a numerical positive value & generate its multiplication table.

```
class Table
{ int n;
  void set (int x)
  {
   n=x;
  }
  void generate()
  {
    if(n<=0)
    {
     s.o.p.(n+" is invalid input, plz enter +ve value");
    }
    else
    {
     for(int i=1; i<=10; i++)
      {
       system.out.println(n+"*"+i+"="+n*i);
      }//
    }//else
  }// generate()
}// BLC
class TableDemo
{
  p.s.v.m (String [] args)
  {
    if (args.length !=1)
    {
     s.o.p(" plz enter one value");
    }
    else
    {
     int x = Integer.parseInt (args[0]);
     Table tb = new Table();
        tb.set (x);
        tb.generate();
    } // else
  } // main()
} // class . ELC
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

1- Q. write a java program which will find sum of 1st N-numbers accept n-value from cmd-prompt; where it should be Positive.

2- Q. write a java program for finding sum of even no. of 1st n-number.

1.

```
Class SumNO
{
 int n,K=0;
 void set(int x)
 {
  n=x;
 }
 void generate()
 {
  if (n==0)
  {
   sop("Enter the valid positive value:");
  }
  else
  {int i=1;
   while (i<=n)
   {K=K+i;
    sop(i+"+");
    i=i+1;
   }
   System.out.println("="+K);
  }//else
 }// generate
}// class-BLC
class SumNum
{
 p.s.v.m(String [] args)
 {if (args.length!=1)
  {sop("enter one value");
  }
  else
  {int x = Integer.parseInt(args[0]);
   SumNO sn = new SumNO();
   sn.set(x);
   sn.generate();
  }
 }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. write a java program which will decide whether the given no. is prime no. or not.

Ans :-

```
// PrimeDemo.java
class Prime
{
 int n;
 void set (int x)
 {
  n=x;
 }
 String decide()
 {
  if (n<=1)
  {
   return (n+" IS invalid Input");
  }
 else
  {int i;
  for( i=2; i<n; i++)
  {int r = n%i;
   if (r==0)
   {
    break;
   } // if
  } // for
  if (i==n)
   return (n+" IS PRIME");
  else
   return (n+" IS not PRIME");
  } // else
 } // decide()
 void disp (String s)
  {
   SOP (s)
  }
} // BLC
class PrimeDemo
{ psvm (String args[])
  { if (args.length != 1)
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
}
 SOP ("enter one value!!");
}
else
 {
  Int x = Integer.parseInt (args [0]);
   prime p = new prime();
     p.set (x);
   String res = p.decide();
     p.disp (res);
   } //else
 }// main()
} // primeDemo - ELC
```

Q. Write a Java Program which convert an ordinary no. into equivalent roman NO's.

Logic Analysis :-

1 - I - while
4 - IV - if
5 - V - if
9 - IX - if
10 - X - while
100 - C - while
90 - XC - if
50 - L - if
40 - XL - if

1000 - M - while
900 - CM - if
500 - D - if
400 - CD - if

```
// ConvertDemo.Java
class Convert
{
  Int n;
  void set (int x)
  {
    n=x;
  }
     void roman()
     {
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
if (n<=0)
{
sopln ("No!! Roman Equivalent");
}
else
{
while (n>=1000)
{
s.o.p ("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;
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
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;
}
}
}
class ConvertDemo
{psvm(String [] args)
 {
  if (args.length != 1)
   {sop("plz enter one value atleast");
   }
  else
   {int x = Integer.parseInt(args[0]);
    Convert cv = new Convert();
     cv.set(x);
     cv.romani()
   } // else
 } //main()
} // class - ELC
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. write a Java program which will print factors of a given no; Accept a no's from cmd prompt dynamically.

```
class Factor
{
  int n, cf = 0;
  void set(int x)
  {
   n=x;
  }
  void generateFactor()
  {
    if(n<=0)
    {
     sop ("invalid input value");
    }
    else
     {
      sop ("_____");
      sop("given number =" + n + "and factor are");
      }
    {
     for (int i=1; i<=n/2; i++)
     {
      int r=n%i;
      if (r==0)
      {
       sop(i);
       cf = cf +1;
      }
     }
     sop ("_____");
     sop ("no. of factor i=" + cf);
     sop ("_____");
    }
  }
}
  class FactorDemo
  {
    psvm (String [] args)
    {
     if (args.length != 1)
      {
       s.o.p.("plz enter one value");
      }
     else
     { Int x = Integer.ParseInt(args[0]);
         Factor f = new Factor();
  } } }    f.set(x); f.generateFactor();
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Constructors in Java

05-Oct-15

Purpose :-

The purpose of constructor in Java is that to initialize the object without placing the default values.

Def:-

A constructor is one of the special member method, which is automatically/Implicitly called by JVM during object creation and whose role is to place our own values without placing default values.

Advantage :-

- If we develop any Java application with the help of constructor then we get the following advantages:-
  (1.) It eliminates in place in default values.
  (2.) It eliminates in calling ordinary method explicitly.

Properties/characteristics/Rules of constructors :

In order to use Constructors in our program we must follow the following rules:-

(a.) Constructors are calling implicitly by the JVM during object creation.

(b.) Constructor name should be similar to class name.

(c.) Constructor willn't return any return type even void also. (If we write any return type which is by default treated as ordinary method)

(d.) Constructors are not static. (because constructors are calling E & E time whenever an object is created

(e.) Constructors never participates in inheritance process (Because every constructors of every class is ment for initializing its own Data member, but not other class Data members)

(f.) The AccessModifier of constructor may or mayn't be Private.
  (1.) If the access modifier of constructor is Private then whose corresponding class object can be created in the same class context but not in the context of other classes.

(2) If the access modifier constructor is not private then the corresponding class object can be created in the context of same class & other class.

- Types of Constructors :-

Depends on creating an object in java, constructors are classified into two types they are —

(1) Default / Parameterless zero argument constructor.

(2) Parameterized constructor.

<1> Default Constructor :-

Def: A constructor is said to be default if and only if it never takes any parameter.

Purpose-

The purpose of default constructor is that to create multiple objects w.r.t. same class for placing same values.

Syntax :-

```
class <clsname>
{
  ----
  <clsname> ( )    // Default constructor
  {
    Block of stat(s)... Initialization
  }
  --------
}
```

Ex-

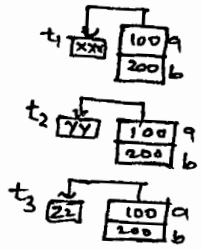
Q. Write a java program which illustrate the concept of default constructor?

```
// TestConstDemo.java
Class Test
{
  int a,b;
  Test()
  {
    SOP ("Test-DC");
    a=100;
    b=200;
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
        S.O.P (" val of a ="+a);
        S.O.P (" val of b ="+b);
    }
} // Test

Class TestConstDemo
{
    P.S.V.M (String [] args)
    {
        Test t1 = new Test();
        Test t2 = new Test();
        Test t3 = new Test();
    }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.



Rules –

1. whenever we create an obj. of any class only with default constructor then defining the default constructor is optional.
   (a) If we don't define default constructor then JVM will execute its own constructor known as System defined default constructor; and it will places default values
   (b) If we define default constructors then JVM will execute Programmer defined default constructor and it places Programmer defined values.

2.> Parameterized Constructor :-

Def :- A constructor is said to be parameterized if & only if it always takes parameters.

Purpose :-
The purpose of parameterized constructor is that "To create multiple object w.r.t. same class for placing different values".

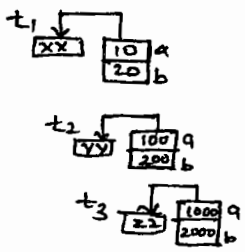
Syntax :-
```
class <clsname>
{
    ____
    <clsname> (list of formal parameter) // PC
    {
        Block of stat(s). - - - . Initialization
    }
    ____
}
```

Ex

Q. write a Java program which illustrate the concept of Para-metrized constructor.

```
// TestPC.java
class Test
{
  int a, b;
  Test (int x, int y)
  {
    S.O.P ("Test - PC");
    a=x;
    b=y;
    S.O.P ("val of a =" +a);
    SOP ("val of b =" +b);
  }
}
  class TestPC
  {
    P S V m (String [] args)
    {
      Test t1 = new Test (10,20);
      Test t2 = new Test (100, 200);
      Test t3 = new Test (1000, 2000);
    }
  }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.



Rule -

(2) When we create an object with Parameterized constructor then it is mandatory to the Java Programmer to define P.C. Otherwise we get compile-time error.

(3) - when ever we create multiple objects with both default parametrized constructors then it is mandatory to Java Programmer to define both default & parametrized constructor otherwise we get compile time error.

→ Overloaded Constructor

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

Signature represents following points:—

a) No. of Parameters
b) Types of Parameters
c) order of parameters

} At least one thing must be different.

ex:-

```
Test t1 = new Test (10, 20);  ———— (1)
Test t2 = new Test (100);  ———— (2)
Test t3 = new Test (10.5f, 20.5f);  ———— (3)
Test t4 = new Test (10, 10.53f);  ———— (4)
Test t5 = new Test (10.8f, 20);  ———— (5)
```

Here, Test(...) is known as OVC.

Note –
constructors can't be overridden because constructors never participates in Inheritence.

→ Object. Parameterized Constructor

- A constructor is said to object Parameterized if and only if it always takes objects as parameters.
- The purpose of object parameterized constructor is that to copy the content of one object into another object where both objects belongs to same type.
- The functionality of object Parameterized constructor is exactly ressembling like copy constructor concept of cpp.

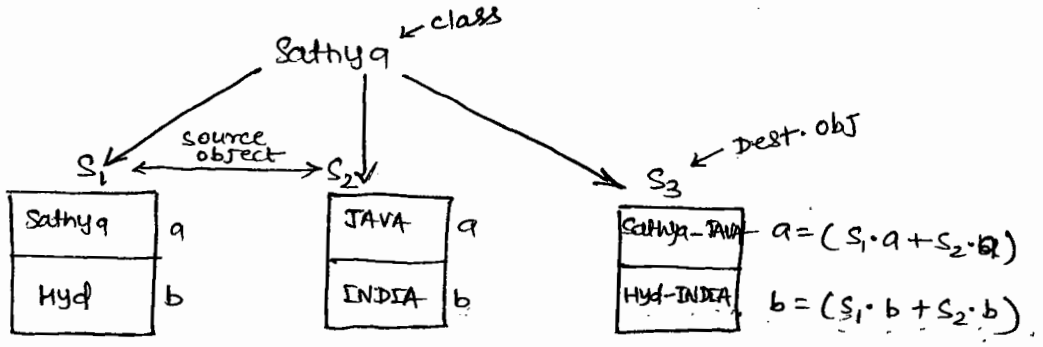
Q. write a Java program which will concatinate contents of two object where each obj. contains two string values.

```
// OPCD.Java
class Sathya
{
  string a, b;
```

```
Sathya (String x, String y)
{
   a=x;
   b=y;
}
Sathya (Sathya x1 , Sathya x2)
   {
    a = x1.a + "-" + x2.a;
    b = x1.b + "-" + x2.b;
   }
   void disp ()
   {
    sop (a+" "+b);
   }
}
class OPCD
{
  psvm (string [] args)
  {
   Sathya s1 =new Sathya ("SATHYA", "HYD");
   Sathya s2 = new Sathya ("JAVA", "INDIA");
   Sathya s3 = new Sathya (s1 ; s2);
    sop ("first source obj. val");
    sop ("_____");
     s1.disp();
    sop ("second source obj. val");
     sop ("_____");
     s2.disp();
     sop ("_____");
      sop ("Dest. obj value");
      sop ("____");
       s3.disp();
       sop ("_____");
  }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

The above program can be representing the following diagram:-



Q. write a Java program which illustrate the concept of default overloaded Parametrized constructor, object Parameterized Constructor and compute addition of two object values where each objects contains two integer values.

```
// TestDemo.java
class Test
{
  int a,b;
  Test()
  {
    sop ("Test-Dc");
    a=1;
    b=2;
    sop (" val of a =" +a);
    sop (" val of b =" +b);
  }
  Test (int x)
  {
    sop ("Test-SPC");
    a=b=x;
    sop ("val of a =" +a);
    sop (" val of b =" +b);
  }
  Test (int x, int y)
  {
    sop ("Test-DPC");
    a=x;
    b=y;
```

```
s.o.p ("val of a =" +a);
 sop ("val of b =" +b);
}
Test (Test x)
 {
   sop ("Test - single OPC");
    a = x.a;
    b = x.b;
    sop ("val of a =" +a);
    sop ("val of b =" +b);
  }
  Test (Test x1 , Test x2)
   {
    sop ("Test - Double - OPC");
    a = x1.a + x2.a;
    b = x1.b + x2.b;
    sop ("val of a =" +a);
    sop ("val of b =" +b);
    }
  } // Test
  Class TestDemo
  {
   psvm (string [] args)
    {
     Test t1 = new Test();
     Test t2 = new Test (100, 200);
     Test t3 = new Test (1000);
     Test t4 = new Test(t3);  ← obj. parameterized constructor
     Test t5 = new Test (t2, t3);  ← double obj. Par. cons.
    }
  }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Points to be Remember:-

- By default E & E class of Java contains System defined default Constructor (Added by JAVA Compiler during compilation time, and executed by JVM during run-time).

- The Default values placed in the object of every class by system define default constructor.

- Programmatically every class of Java can contain two types of constructor they are –
  (a) single programmer defined default constructor.
  (b) Multiple parametrized constructors (known as overloaded constructors)

- In real world application before using any class or Interface it is highly recommended to the programmer to view a profile of class/Interface.
- A profile is nothing but details about data members, methods, constructors etc.

- To view the profile we use a tool called Javap which is present in Java.Home\bin directory.

  syntax:-

Javap fully Qualified Name of a class/Interface ↵

- Javap tool to be applied only on .class file after compilation process

ex:-
- Javap Test (Here we can view the profile of Test class)

Note - If we aren't initializing data members of class by using constructors then those data members will be initialized by system defined default constructor with default value.

* Data Encapsulation:-

The process of hiding the information from external users to prevent the unauthorized modification is known as Data Encapsulation.

- To Hide the data / information from external users we use "Private" in other word private features of a class never participate in profiling process.

Note:-

1. Some of the classes of Java doesnot contains constructors (False/True) _ false _ (Because every class of Java contains constructors developers of the class made those constructor as private.

* Data Abstraction

- Definition:-

The process of retreiving or abstracting the essential detail without considering hidden details : is called ; data Abstraction.

→ Reading the data from keyboard

- To read the data from keyboard we have a predefine class called Scanner which is present in predefine Package called Java.util.* and this class is available from JDK 1.5 version onwards.
- Reading the data from KBD is nothing but creating an object of Scanner class.
- With Scanner class object we can read only one value at a time but not multiple values.

Profile of Java.util.Scanner

Constructor

① Scanner (InputStream)

Instance ()

② Public xxx nextXxx().

③ Public String nextLine()

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- Constructor is used for creating an object of Scanner class by passing an object of inputStream class.

- An object of inputStream called in created as a static data member in an another predefine class called System.

```
Scanner s = new Scanner(System.in);
```

Here 's' can read actual data

Provides an Env. to read the Data from keyboard

- In method (2) xxx represents any fundamental data type. method (2) is used for reading fundamental type values from the keyboard.

- method (3) is used for reading any type of data from KBD in the form of String type.

Q. Write a Java program which will read two integer value from keyboard & compute their sum.

```
// key.java
import java.util.Scanner;

class key
{
 Public static void main (String [] args)
 {
   Scanner s = new Scanner(System.in);
   SOP("Enter the first int val=");
     int n1 = s.nextInt();
   SOP("Enter the Second int val=");
     int n2 = s.nextInt();

     int n3 = n1 + n2;
   SOP("Sum("+n1+" , "+n2+") = "+n3);
 }
}
```

Q. write a Java program which will accept student information such as number, name, collegeName, marks ?

```
// StudentData.Java
import Java.util.Scanner;
class StudentData
{
 PSVm (String [] args)
{
  Scanner s = new Scanner (System.in);
   SOP ("Enter the student number:");
    int Sno    = Integer.parseInt (s.nextLine());
   SOP ("Enter the student name:");
    String Sname = s.nextLine();
   SOP ("Enter the college name:");
    String Cname = s.nextline();
   SOP ("Enter the student marks:");
     Float marks = Float.parseFloat (s.nextline());
   SOP ("_________");
   SOP ("student Detail");
   SOP ("_________");
   SOP ("student number:" + Sno);
   SOP ("student name:" + Sname);
   SOP ("student college name:" + Cname);
   SOP ("Student marks:" + marks);
   SOP ("_________");
   SOP ("_________");
  }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. write a java Program which will read no. of no's of type integer & compute there sum.

```
// SumNum.Java
import java.util.Scanner;
class SumNum
{
  psvm (String [] args)
  {
    Scanner S = new Scanner (system.in)
    sop (" Enter How many no's u want add");
    int S1 = 0;
    sop ("_____");
    for ( int i=0 ; i<nos ; i++)
     {
      sop (" Enter" +(i+1) +" value");
      int n = Integer.parseInt( S.nextLine());
       S1 = S1+n;
     } // for
    sop (" _____");
    sop (" sum =" +S1);
    sop (" _____");
  }
}
```

* Performing Mathematical Operation in Java

- In Java Programming to Perform any mathematical operation we use a predefine class called Java.lang.Math
- All mathematical operation are given in the form of Predefine static method.

Q. write a Java Program which will calculate (x^n).

```
// PowDemo.Java

import Java.util.Scanner;
      class PowDemo
      {
        psvm (String [] args)
        {
```

```
Scanner s = new Scanner (system.in);
 sop (" Enter Base (x):");
 int x = Integer.parseInt( s.nextLine());
 sop ("Enter power n:");
 int n = Integer.parseInt(s.nextLine());
 sop (" pow ("+x+"; "+n+") =" + Math.pow(x,n));
 }
}
```

Q. write a Java program which will accept an integer value & find its square root ($\sqrt{x}$).

```
// Sqrt.Java
import Java.util.Scanner;
 class Sqrt
 {
  p.s.v.m (String [] args)
  {
   Scanner s = new Scanner (system.in);
    sop (" Enter the value (x):");
    int x = Integer.parseInt (s.nextLine());
     sop (" square of "+x+" is" + math.sqrt(x));
   }
  }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Static Blocks in Java

Purpose –

- The purpose of static Block is to initialize the static data members.
- Static Blocks will be executed only once when the class is loaded in main-memory.
- The purpose of constructor is to initialize the instance data members and it will be called by JVM E&E time whenever an object is created, whereas Static block is meant for initializing static data member only once and it will be executed by JVM when the class is loaded in main memory.
- Inside the static block we should not initialize instance data member.
- If we use main method & static ~~method~~ block in program; then JVM will execute static block first and main method only once.
- Writing static block is optional (when the static DM is not present in the class).

Syntax of static block :–

```
Static
{
  Block of Statement (s) —— Initializing static data member
}
```

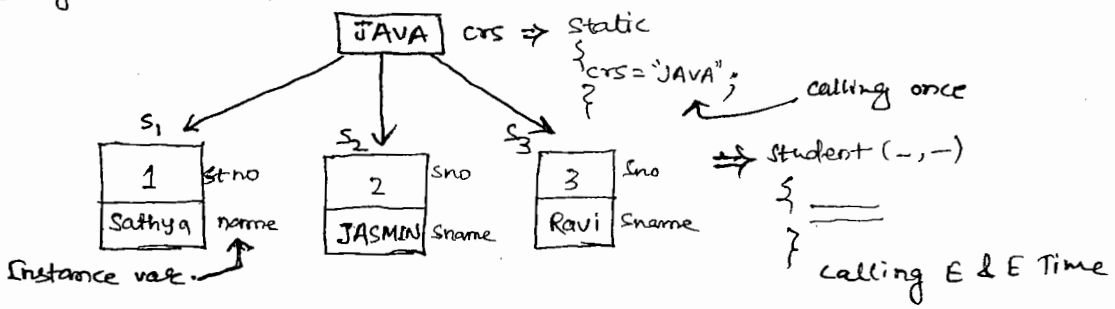
- Static block must be written within the class but not inside the method definition.
- In a Java Program one can write Multiple static blocks and Internally of them are treated as single static block & they execute only once in the order in which we write.

Q. write a Java Program which illustrate the Concept of static block for initializing the static data members.

```
//StudentDemo.Java
class Student
{
  int no;
  String name;
  Static String crs;
  Student (int Sno, String Sname)
  {
     no = Sno;
     name = Sname;
// crs = cr ⇒ Not recommended to
     SOP ( Sno+ "," Sname+ ", "+crs);
  }
  Static
  {
     //Stno = 10; error
     crs = "Java";
  }
}
class StudentDemo
{ PSVM (String [] args)
  {
    Student S1 = new Student (1, "Sathya");
    Student S2 = new Student (2, "Raju");
    Student S3 = new Student (3, "Jasmin");
  }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

The diagramatic representation of the above program is shown as—



Q. can we execute Java prog without main(-)

- Yes, we can execute Java prog without main (-) up to JDK 1.6 version.
- we can't execute Java program without main(-) even though we write static block from JDK 1.7 version onwards.
- In otherwords, from JDK 1.7 onwards if we write Java program without main(-) we get compile time error.

Code:-
ex -

```
// SBEX1.Java
Class SBEX1
{
  Static
  {
    SOP ("static block - SBEX1");
    System.exist(0);
  }
}
```

O/P → static block - SBEX1 // valid up to JDK 1.6 but if we compile above prog. from JDK 1.7 onwards we get compile-time-error.

Q. write a Java program which illustrate static block & main(-).

```
// SBEX2.Java
Class SBEX2
{
  PSVM (String [] args)
  {
    SOP ("i am from main()");
  }
  Static
  {
    SOP ("SBEX2 - static block");
  }
}
```

Output :- SBEX2 - Static block
i am from main()

Q. write a java program which illustrate multiple static block in a class.

```
// SBEX 3.java
class SBEX3
{
  static
  {
    sop ("SBEX3 - static block 1");
  }
  psvm (string [] args)
  {
    sop ("i am from main ()");
  }
  static
  {
    sop ("SBEX 3- static block 2");
  }
}
```

O/P :- SBEX 3 - static block 1
SBEX 3 static block 2
i am from main ()

Q. write a java program which illustrate static block & constructor in a business logic class.

```
// SBEX4.java
class Test
{
  static
  {
    sop ("test - static block");
  }
  Test ()
  {
    sop ("Test -DC");
  }
}// Test
class SBEX4
{
  psvm (string [] args)
  {
    sop ("i am from main ()");
    Test t1 = new Test ();
    Test t2 = new Test ();
  }
}
```

Output

i am from main ()
test - static block
Test - DC
Test - DC

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. write a java program which illustrate static blocks and constructors in BLC & ELC -

```
//SBEX5.java
Class Sample
{
 Static
  {
   system.out.println ("Sample- Static block");
  }
 Sample (int x)
  {
   sop ("Sample -pc");
  }
} // Sample-BLC
 class SBEX5
 {
  SBEX 5()
  {
   sop ("SBEX5-DC");
  }
  static
  {
   sop ("SBEX5 - static block");
  }
  psvm (String [] args)
  {
   sop ("i am from main()");
   Sample s1 = new Sample(10);
   Sample s2 = new Sample (20);
   SBEX5 s6 = new SBEX5();
   SBEX5 s7 = new SBEX5 ();
```

Output:-

```
SBEX5 - static block
i am from main()
Sample - static block
Sample - pc
Sample -pc
SBEX5 -DC
SBEX5 -DC
```

Q. Write a java program which illustrate the important of system - define static block.

```
Class SBEX7
{
  static int a=10;
}
```

When we compile the above program the compilation env. will add system defined static block and it which the above static variable initialization will be placed.

Q. When we get system-defined static block?

When we write/use static variable initialization in any class of java then the compilation environment will automatically add system defined static block.

Instance Block in Java

Purpose:- The purpose of Instance Block is that initializing Instance DM of a class. The purpose of Instance Block is exactly similar to programmer defined default constructor.

- Instance Block will execute E & E Time, before executing constructor and after executing static Block. When obj. is created;

- One can write multiple Instance Blocks and internally they will be treated as one instance Block only and it will execute in the order which will be write.

Syntax -

```
{
    Block of Stat(s) ------ Initializing IDM.
}
```

- Instance Block we will write within the class and we shouldn't write within the methods.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. write a Java program which illustrate the concept of instance block, static block, constructor & main(-) in an ELC.

```
// IBEX1.java
class IBEX1
{
  IBEX1()
  {
   sop("IBEX1 — DC");
  }
  {
   sop("IBEX1 - Instance Block");
  }
  static
  {
   sop("IBEX1 - static block");
  }
  psvm(String[] args)
  {
   sop("I am from main()");
   IBEX1 io1 = new IBEX1();
   IBEX1 io2 = new IBEX1();
  }
}
```

O/P :- IBEX1 - static block
I am from main()
IBEX1 - Instance block
IBEX1 - DC
IBEX1 - Instance block
IBEX1 - DC

Q. write a Java program which illustrate the concept of Instance DM in the BLC with the help of instance block.

```
// IBEX2.java
class Sample
{
 int a,b;
 {                                  // Instance Block
  sop("Sample - Instance Block");
   a=1;
   b=2;
```

```
    sop ("val of a ="+a);
    sop ("val of b ="+b);
  }
} //Sample - BLC

Class IBEX2
 {
  psvm (String [] args)
  {
   sop ("I am from main []");
    Sample s1 = new Sample ();
    Sample s2 = new Sample ();
   }
 }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Output:-

```
I am from main ()
Sample - Instance Block
 val of a = 1
 val of b = 2
Sample - Instance Block
 val of a = 1
 val of b = 2
```

Q. Write a Java program which illustrate static block, instance Block, constructors in BLC?

```
//IBEX3.java
class Sample
 { int a,b;
   Static
   {
    sop ("Sample - static block");
   }
   Sample (int x, int y)
   {
    Sop("Sample - PC");
    a=x;
    b=y;
    Sop ("val of a =" +a);
    Sop ("val of b = "+b);
   }
```

```
{
  sop ("sample-Instance block");
  a=1;
  b=2;
  sop ("val of a ="+a);
  sop ("val of b ="+b);
}
}
class IBEX3
{
  psvm (string [] args)
  {
    sop ("I am from main ()");
    Sample S1 = new Sample (10,20);
  }
}
```

O/P :-	I am from main ()
	Sample — static block
	Sample — Instance block
	value of a = 1
	value of b = 2
	Sample — pc
	Val. of a = 10
	Val. of b = 20

Q. write a java program which illustrate multiple instance block, static blocks, overloaded constructor in BLC & in ELC ?

```
// IBEX4.java
class Test
{
  Test (int x)
  {
    sop ("Test-pc");
  }
  }
  sop ("Test-IB-1");
  }
  static
  {
    sop ("Test-SB-1");
  }
  Test()
  {
    sop ("Test-DC");
  }
```

```
{
 SOP ("Test - IB-2");
}
Static
 {
  SOP ("test- SB-2");
 }
} // Test - BLC
Class IBEX4
{
 {
  SOP ("IBEX 4 - IB-1");
 }
  Static
  {
   SOP ("IBEX4- SB-1");
  }
  IBEX4 (int x)
  {
   SOP ("IBEX4- PC");
  }
  PSVM (String [] args)
  {
   SOP ("i am from main()");
    IBEX4 O4 = new IBEX4 (10);
    Test t1 = new test (10);
    Test t2 = new test ();
  }
} // IBEX-4 — ELC
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

O/P:-

```
IBEX4 — SB1
I am from main()
IBEX4 --- IB1
IBEX4 --- PC
Test ---- SB1
Test ---- SB2
Test ---- IB1
Test ---- IB2
Test ---- PC
Test ---- IB1
Test ---- IB2
Test ---- DC
```

Note :-

- A class of Java contains five Parts they are —
  - Data member
  - methods
  - Constructor
  - Static Block
  - Instance Block

-

" this "

- 'this' is one of the Keyword / Implicit object created by Java Compilation Environment & available in every Java program for two purposes.

(1.) It always points to current class object ('this' contains current class object refrence).

(2.) Whenever the formal parameters & data member of class are same then JVM gets an ambiguity. (Ambiguity - No clearity betn multiple duplicate things) In order to differentiate betn data members & formal parameters, the data members of the class must be preceded by 'this' otherwise they will be treated as formal parameters.

Syntax :-

this.currentClass DataMember Name

Q. write a Java Prog. which illustrate the concept of 'this'.

```
// ThisEx.java
Class Test
{
  int a, b;
  Test (int a, int b)
  {
this.a = a;
this.b = b;
```

// we are assigning formal Parameter val. to the data members of current class

```
this.a = this.a +1; // here we are incrementing the val of
this.b = this.b +1;   current class data members.
    a = a+2;      // here we are incrementing
    b = b+2;         Formal parameter values.
    Sop ("val of (Formal Params) a =" +a); // here a, b refers
    Sop ("val of b (Formal Param) =" +b);      Formal Parameters
}
  void disp ()
  {
    Sop ("val of a =" + this.a); // here writing 'this' is
    Sop ("val of b =" + this.b);    optional
  } // disp()
} // Test
Class ThisEx
{
  P.S.V.M (String [] args)
  {
    Test t1 = new Test (10,20);
        t1.disp();
  }//main()
} // ThisEx - ELC
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Consider the following statement -

```
obj1.methodName(obj2);
```

Here obj1 is called source/invoking object &
obj2 is called destination/Target object
When the above statement is executed, in the method definition source object values are reffered as

```
this. data member of source object
```

& destination object values are reffered as

```
Formal object . Data
```

Q. write a java program which illustrate the above rules.

```
// ThisEx1.java
class Test
{
  int a, b;

    Test (int a, int b)
    {
      this.a = a ;
      this.b = b ;
    }
    void disp (Test x) // here x are called formal obj.
  { sop ("The val of a =" + this.a);
    sop ("The val of b =" + this.b);
    sop ("The val of a =" + x.a);
    sop ("The val of b =" + x.b);
  }
}
  Class      ThisEx1
  {
    psvm (String [] args)
    {  Test t1 = new Test (10, 20);
       Test t2 = new Test (100, 200);
            t1.disp (t2) ; // here t1 is called
                             source invoking object,
                             t2 called destination obj.
    }
  }
```

Q. write a Java Program which will compute sum of content of two objects where each objects contains two integer values.

```
// ObjSum.Java
import java.util.Scanner;
Class Test
{
  int a, b;
  Test (int a, int b)
   {
    this.a = a;
    this.b = b;
   }
  Test ()
  {
   a=b=0;
  }
  Test Sum (Test x)
   {
    Test t33 = new Test();
    t33.a = this.a + x.a;
    t33.b = this.b + x.b;
      return t33;
   }
   void disp();
   {
    sop ("val of a =" +a);
    sop ("val of b = " +b);
   }
}// Test-BLC
 Class objSum
 {
   PSVM (String [] args)
   {
 // accept dynamic values
     Scanner s = new Scanner (System.in)
      sop ("Enter first val. for first object");
      int x1 = Integer.parseInt (s.nextLine());
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
(OR) // void sum (Test x1, Test x2)
     // {
     //  this.a = x1.a + x2.a;
     //  this.b = x1.b + x2.b;
     // }
```

```
SOP ("Enter second val for first object");
int x2 = Integer.parseInt(s.nextLine());
SOP ("Enter first val for second object");
int x3 = Integer.parseInt(s.nextLine());
SOP ("Enter second val for second object");
int x4 = Integer.parseInt(s.nextLine());

// create the object with dynamic values
        Test t1 = new Test(x1, x2);
        Test t2 = new Test(x3, x4);
        Test t3 = new Test();
        // t3 = t1 + t2  error
        t3 = t1.sum(t2);   OR // t3.sum(t1, t2);
        SOP ("t1 values:");
        SOP ("_____");
        t1.disp();
        SOP ("_____");
        SOP ("t2 values:");
        SOP ("_____");
        t2.disp();
        SOP ("_____");
        SOP ("Object sum");
        SOP ("_____");
        t3.disp();
        SOP ("_____");
    }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Note:- Java programming never supports operator overloading because it support static Polymorphism.

- The aim of operator overloading can be obtained by simple concept of methods in Java.

Factory Method.

- A factory () is one whose return type is similar to class name, in which class it present.

- The purpose of factory () is to create an object without using "new" operator.

- We have two types of factory method –
  - Instance factory ()
  - Static factory ()

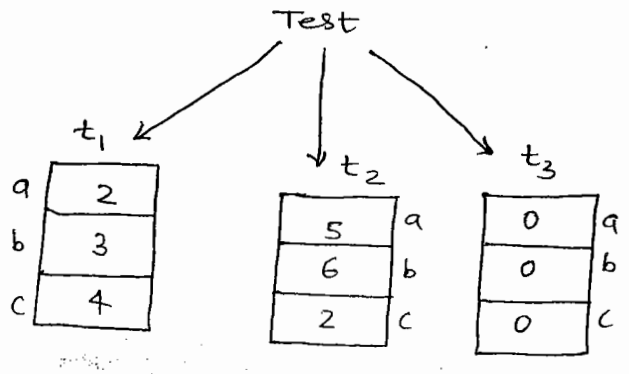
ex:-

Test

Test sum (Test)
Test sum (int, int, int)
} Factory Method

Q. Write a Java Program which will multiply contents of two object where each objects contains three int values.

Plan:-



$t_3 = t_1 * t_2$; // error

1. Solⁿ :– t_3.mul (t_1, t_2)
2. Solⁿ :– $t_3 = t_1$.mul (t_2)

Prog:-

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Inheritance

21-Sep-15.

- Inheritance is one of the distinct principle of oop's and with this principle we can develop reusable application.

- **Definition:-**

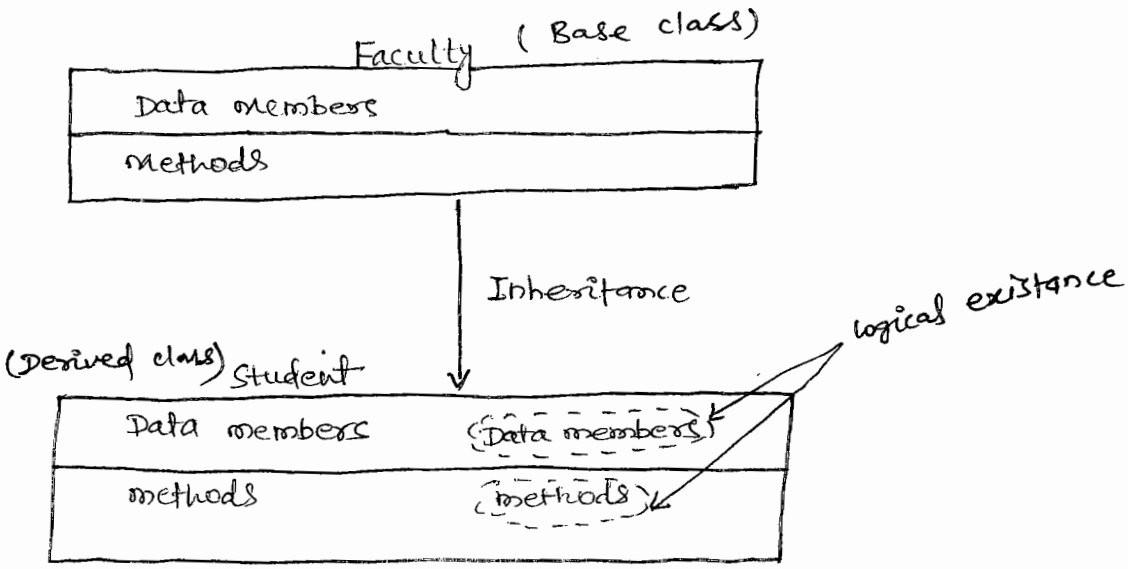
The process of getting the data members & methods of one class into another class is known as inheritance.

- In inheritance process the class which is giving data members & methods is called Base/super/Parents class. where as the class is taking data members & Methods is called Derived/sub/child class.

- The data members & Methods of class are called features inheritance based Appl^n always follows Logical memory management. It says Base class features are available in the context of derived without taking any memory space. In other words logical memory management says-

"feeling that Base class features are available in derived class but in reality they don't take any memory space".

- The following diagram use ~~used~~ about inheritance & Logical memory mangement.



The concept of inheritance are also known as sub classing, derivation, reusability or extendable classes

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Adv. of Inheritances —

If we develop any Java application with the concept of inheritance then we get following adv :—

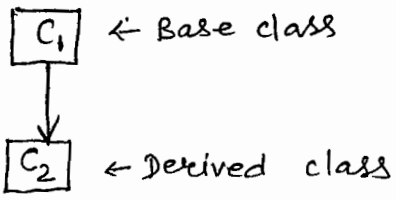
1. application development time is Less.
2. Application memory space is Less.
3. Appl^n execution Time is less.
4. Appl^n performance is enhanced (Improved).
5. Redundency of code is minimized.
6. Able to get the slogen of Java (i.e. WORA).
   write once run Anywhere.

Types of inheritances / Reusable Technique :-

- A pattern or model which makes us to understand how to inherit the features from Base class to Derived class is called type of inheritance or reusable technique.

- In Java programming we have 5 types of inheritances; they are :-
  1. Single inheritance
  2. Multilevel inheritance
  3. Hierarical inheritance
  4. Multiple inheritance
  5. Hybrid inheritance

1. **Single inheritance** :-
   in this inheritance there exist single Base class & single derived class.



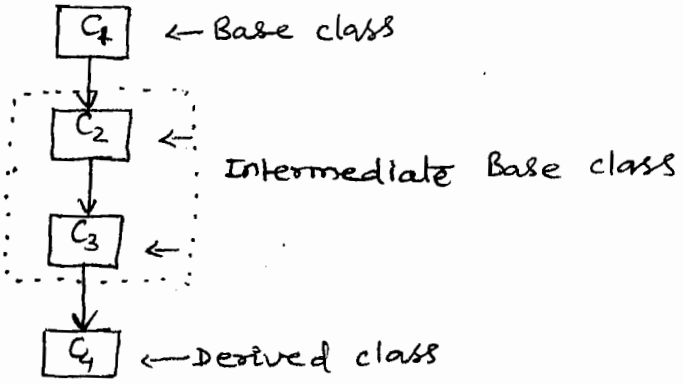
Inheritance Path –

$$C_1 \rightarrow C_2$$

2. multilevel inheritance :-

In this inheritance there exist single Base class, single derived class and multiple intermediate Base classes.

Diagram :-



Path –

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

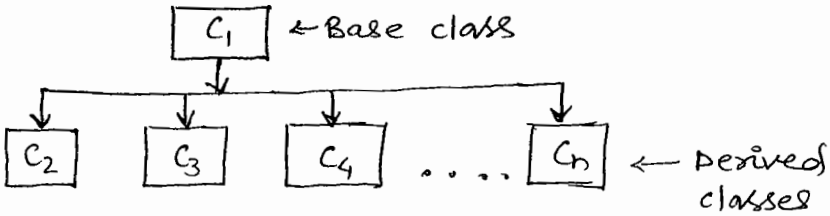
Note :-

An I.B.C is one class in one context it acts as base class and in another context same class acts as Derived class.

3. Hierarichal inheritance :-

In this inheritance there exist single Base class & multiple Derived classes.

Diagram :-



Path :-

$C_1 \rightarrow C_2$

$C_1 \rightarrow C_3$

$C_1 \rightarrow C_4$

$\vdots$

$C_1 \rightarrow C_n$

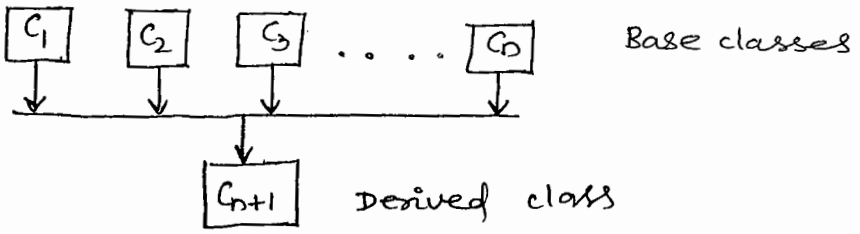
The above 3-types of inheritances supported in Java By Both classes & interface.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

4. multiple inheritance:-

In this inheritance there exist multiple Base classes, and single derived class.

Diagram :-



Path -

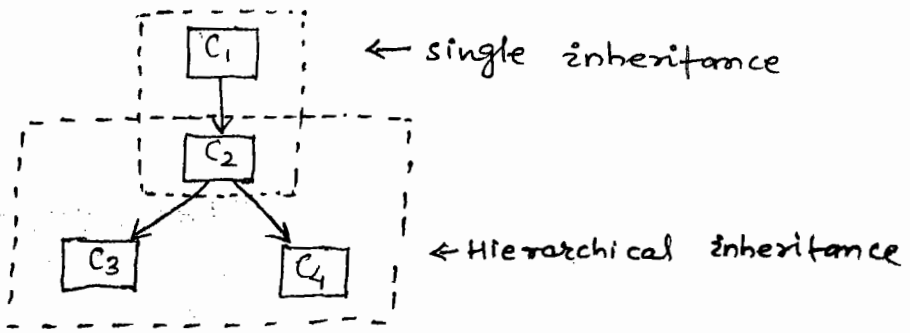
$C_1 \rightarrow C_{n+1}$
$C_2 \rightarrow C_{n+1}$
$C_3 \rightarrow C_{n+1}$
⋮
$C_n \rightarrow C_{n+1}$

multiple inheritance is not supported in Java by the concept of classes due to Ambiguity Problem.
But it can be supported in Java by the Concept of interfaces.

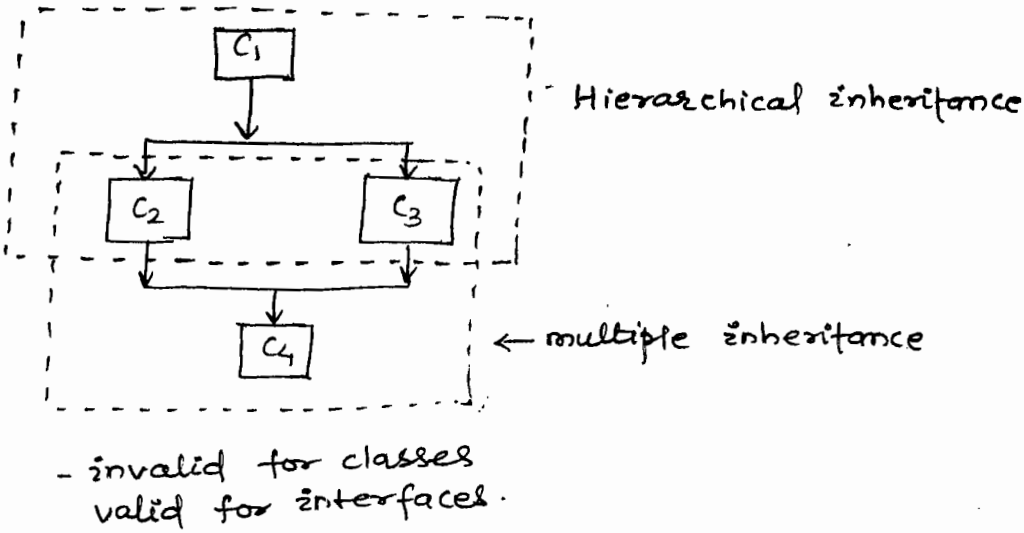
5. Hybrid inheritance

Hybrid inheritance = combination of any available inheritance Type.

In the combination if one of the combination is multiple inheritance then the entire combination (Hybrid inheritance) is not supported by classes; but it can be support interfaces concepts.

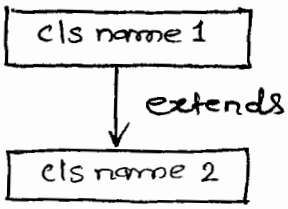


valid by both class & interfaces.



Inherating the features from Base class to derived class :-

Diagram :-



Syntax :-

```
class <clsname 2> extends <clsname 1>
{
    var declaration;
    method definition;
}
```

explanation :-

1.> clsname 1 & clsname 2 represents name of the base & derive class.

2.> "Extends" is a keyword used for inherating the features of base class into derived class. + it provides richest functionality to derived class.

3.> Rule:- one derived class can extends only one Base class Because Java Prog. never supports multiple inheritance (But it can be supported by the concept of interfaces)

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

4.) whenever we develop any inheritance Based Appl^n, it is always recommended to create an object of Bottom most derive class because it inherits the features of Top most Base class & Intermediate Base class.

5.) whenever we create an object of Bottom most derive class, first we get the memory space for data members of top most base class, Secondary get the memory space for data member of intermediate base class & last get the memory space for data member of Bottom most derive class. Hence, All the features of Top most base class & intermediate base class are available logically in the context of bottom most derive class.

6.) If we don't want to give the features of base class to derive class then the definition of base class must be made as <u>final</u>; Hence <u>final base classes never participate in inheritence.</u>

final features of base class can be inherited into derived class but they can't be changed in the context of derived class.

7.) If we don't want to give some of the features of base class to derive class those features must be made as private Hence; Private features of Base class never participate in inheritence or never visible/accessible in the context of derived class.
Java doesn't contain just private classes; but inner private class can be allowed.

8.) In Java Programming data members & methods can be inherited by default but Constructors, Static block, Instance block will not Participate in inheritance.

9.*) <u>Scope of Basic class object</u>:
An object of Base class contains an info. about Base class features but whose obj never contain info. about special features of sub-class/derived class.

10.) for each & every class in Java there exist an implicit Predefine superclass called Java.lang.object because it Provides garbage collection facility (finalize()).
To its sub classes for collecting unused memory space for improving the Performance of Java based application.

✡ write a Java Program which illustate the concept of inheritance :-

```
// Inhprog1.java
class BC
{
 int a;
 void disp()
 {
  System.out.println("base - disp()");
 }
} //BC
class DC extends BC
{
 int b;
 void show()
 {
  System.out.println("derived - show()");
  a=10;
  b=20;
  System.out.println("val of a from BC =" +a);
  System.out.println("val of b from DC =" +b);
 }
} //DC
class Inhprog1
{
 public static void main (String[] args)
 {
  System.out.println("w.r.t. DC");
  System.out.println("_____");
  DC do1 = new DC();
   do1.disp();
   do1.show();
  System.out.Println("_____");
  System.out.Println("w.r.t. BC");
  System.out.println("_____");
   BC bo1 = new BC();
   bo1.disp();
   System.out.println("_____");
 }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

* Types of relationships in Java :-

Relationships in Java always makes us to understand how to use the features of one class to another class.

In Java Programming we have three types of relationship.

1. IS-A relationship
2. has-A relationship
3. User-A relationship.

1. is-A relationship :-

In this relationship one class obtain some features of another class by using inheritance concepts

The relationship betn base & derived class is called IS_A relationship.

ex:-

```
class C1
{
  ___
  ___
}
class C2 extends C1
{
  ___
}
```

IS_A

Here the relationship betn C_1 & C_2 is "IS_A".

In is_A relationship we always follows logical memory manag.
The Limitation of is_A relationship is that unable to eliminate amiquity Problem, which are occuring in multiple inhertance. and Hybrid inheritance (Diamond inheritance problem)
This problem can be solved by using "has_A" relationship.

2. has_A relationship -

In this relationship an obj of one class is created as data member in the context of another class.

ex:-
```
class C1
{
  ___
  ___
}
```

```
class C2 <-- has_A --> 
{
  C1 O1 = new C1();
  ____
  ____
}
```

Here the relationship betn C_1 & C_2 is "has_A"

- "has_A" relationship always follows physical memory management.
- The advantage of "has_A" relationship is that eliminating ambiguity problem occuring in multiple inheritance & Hybrid inheritance (Diamond problem).

consider the following invalid code segment

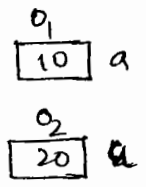
invalid ex:-

```
class C1
{
  int a=10;
}

class C2
{
  int a=20;
}

class C3 extends C1, C2 // invalid (is_A)
{
  ____
  ____
}
```

The above code can be re-written by using has_A relationship.

valid ex:-

```
class C3
{
  C1 O1 = new C1();
  C2 O2 = new C2();
  int b;
  ____
  ____
  b = O1.a + O2.a;
}
```



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

The relationship betn C_1, C_2 with C_3 is called has_A.

3. User_A relationship

In this relationship a method of one class is using an object of another class.

ex:-

```
class C1
{
}
class C2
{
  void disp ()
  {
    C1  O1 = new C1();
  }
}
```

User_A relationship.

Conclusion:-

Hence;

User_A relationship provide method Level scope, has_A rel^n provides class level scope & is_A relationship provides application level scope.

NOTE -

(1.) The default relationship in Java is "is_A" relationship; because for E & E class of Java there exist an implicit predefine superclass called Java.lang.object.

(2.) The universal example for has_A relationship is that System.out (Here out is an object of print stream class and it is created as a static data member and an another predefine class is called System.

(3.) The universal example for user_A relationship is that Each & Every execution Logic method of execution Logic class is using an object of Business Logic class.

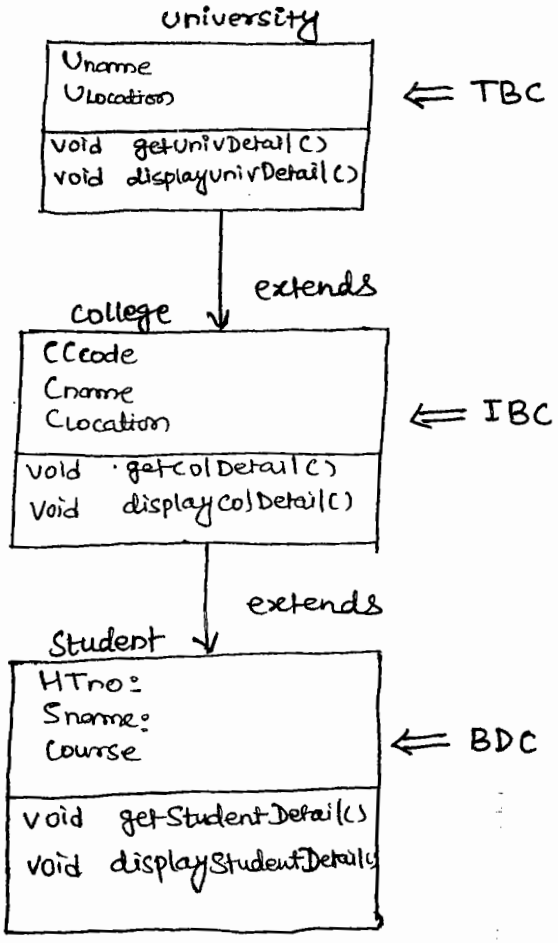
SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Develop a Java Application for the following -

Let us assume that a universal university which contain university name & location. Obtain university details from keyboard & display.

Let us assume that exist a college which is associated with an university where college contain college code, college name & location. Accept college detail along with university details and display.

Let us assume there exist a student which contain hall ticket No, student name & type of course pursuing. Accept student details along with college and university detail and display complete information about student.



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
import java.util.Scanner;
class univ
{
 private String Uname;
 private String ULoc;
 Scanner s = new Scanner (System.in);
 void getUnivDet()
 {
  System.out.println ("Enter Univ Name:");
  Uname = S.nextline();
  System.out.println ("Enter Univ Loc:");
  uloc = s.nextline();
 }
 void dispUnivDet()
 {
  system.out.println ("University detail");
  system.out.println ("_____");
  system.out.println ("Univ Name:" + Uname);
   SOP ("Univ Loc:" + ULoc);
   SOP ("_____");
 }
}  // Univ - TBC
class college extends univ
 {
  private int cc;
  private String Cname, Cloc;
  void getcolDet()
  {
   System.out.println ("Enter college code:");
    cc = Integer.ParseInt (s.nextline());
    SOP ("enter college name:");
     Cname = S.nextline();
    SOP ("enter college loc:");
     Cloc = S.nextline();
     getUnivDet();
   }
   void dispcolDet()
```

```
{
   dispUnivDet();
   sop ("college Detail");
  sop ("_____");
   sop ("college name:"+Cname);
   sop ("college code:"+Ccode);
   sop ("college Loc:"+Cloc);
   sop ("_____");
   }
} // college - IBC
Class student extends college
{
 int htno;
 String Sname;
 String crs;
 student getStuDet().
 {
  sop ("enter student HTno:");
  htno = Integer.ParseInt(s.nextline());
  sop ("enter student name:");
  Sname = S.nextline();
  Sop ("enter student course:");
  crs = S.nextline();
   getcolDet();
  return this; // here we are returning current class obj
                 and 'this' represents current class obj.
 }
 void disStuDet()
 {
  dispcolDet();
  sop ("Student Detail");
  sop ("_____");
  sop ("student Htno:"+htno);
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
        sop ("student name:" + sname);
        sop ("student course:" + crs);
        sop ("_____");
    }
  } // student - BDC
  class Inhprog4
  {
    PSVM (String[] args)
    {
    // By using Name oriented object
          student s = new student();
          student s1 = s.getstudDet();
          s1.dispstudDet();
          //OR
    // By using Name Less object
        new student().getstudDet().dispstudDet();
    }
  }
```

Q. write a Java program which listed the concept of nameless object.

OR

write a Java program for returning an object of current class.

```
// NamelessDemo.java
  class Sathya Tech
  {
    Sathya Tech show() // Instance factory method
    {
      sop ("I am from show()");
      return this;
    }
```

```
Sathya Tech disp()
{
  SOP ("I am from disp ()"); // Instance factory method
  return this;
}
void write()
{
  SOP ("I am from write ()");
}
} // sathya Tech
class NameLessDemo
{
  public static void main (String [] args)
  {
    new sathya Tech().show().disp().write();
  }.
}
```

Q. Develop a Payroll application for the following Problem Statement-

" There exist a company which contain trade Licence No., Company Name & type of operations. Accept the company detail & display.

A company contains a Permanent employee where an employee contains Number, Name, Designation & salery.

calculate the following

additions (GIVE)

DA → 20% of basic sal
TA → 10% of basic sal
CCA → 5000₹
MED All → 10% of basic sal

deduction (Take)

PF: 5% of basic sal
GPF: 5% of basic sal
LIC: 2% of basic sal

Gross sal = (basic sal + DA + TA + CCA +

Display the employee Pay slip along with company detail.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

There exist an outsourcing employee for working a company, name, number and no. of working days & wage per day.

Total salery of an outsourcing empl. =
no. of days present * wage per day

print outsourcing employee pay slip along with company detail.

```
// payslips.java
import java.util.Scanner;
class company
{
private String Cname, trlno, oper
    Scanner s = new Scanner (system.in);
    void getCompDet()
    {
     sop ("enter company Name:");
      Cname = s.nextline();
      SOP ("Enter company Trade Licence NO:");
      trlno = s.nextline();
      sop ("Enter Type operations in a company:");
      Oper = s.nextline();
    } // getCompDet()
    void dispComDet()
    {
      sop ("= = = = = = = = = = =");
      sop ("company Detail");
      SOP ("company Name:" + Cname);
      sop ("Trade Licence No:" + trlno);
      sop ("Operation Type:" + oper);
      sop ("= = = = = = = = = = = =");
    }
} // company - TBC
```

```
class perEmp extends company
{
  int empno;
  string ename; string desig;
  float bsal, da, ta, cca, ma, pf, gf, lic, gsal;
  void getPerEmpDet()
  {
   sop ("Enter Permanent Emp Number :");
   empno = Integer.ParseInt(s.nextLine());
   sop("Enter Permanent Emp Name:");
   ename = s.nextLine();
   sop ("Enter Permanent Emp Desig:");
   desig = s.nextLine();
   sop ("Enter permanent Emp basic sal :");
   bsal = Float.ParseFloat(s.nextLine());
  // cal values of additions
    da = bsal * 0.2f;
    ta = bsal * 0.1f;
    cca = 500.00f;
    ma = bsal * 0.1f;
  // cal values of deductions
     pf = bsal * 0.05f;
     gpf = bsal * 0.05f;
     lic = bsal * 0.02f;
  // cal gross sal
   gsal = (bsal + da + ta + cca + ma) - (pf + gpf + lic);
   }
   void dispPerEmpPaySlip()
   {
    sop ("_____");
    sop ("permanent Emp sal Detail");
    sop ("_____");
    sop ("Emp NO:" + empno + " Name :" + ename);
    sop ("Emp Designation " + desig);
    sop ("Basic sal = " + bsal);
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
        SOP ("DA = " +da + "TA  "+ta);
        SOP ("CCA = "+cca+"Med Allow ="+ma);
        SOP ("PF = "+pf+" GPF = "+gpf+" LIC ="+lic);
        SOP ("_____________________");
        SOP ("Grass Salary = "+gsal);
        SOP ("_____________________");
    }
} // perEmp - BDC

class outsourceEmp extends company
{
    int oempno;
    String oename;
    int nodw, wages; int tsal;
    void getOutEmpDet()
    {
        system.out.println("Enter outsource Emp Number:");
            oempno = Integer.ParseInt (s.nextline());
        System.out.println("Enter Outsource Emp Name:");
        oempname = S.nextLine();
        SOP ("Enter no. of days outsource Emp worked :");
        nodw = Integer.ParseInt (s.nextline());
        SOP ("Enter wages outsource Emp Per day :");
        wages = Integer.ParseInt (s.nextline());
        tsal = nodw * wages;
    }
    void dispoutEmp Pay slip ()
    {
        SOP ("_____________________");
        SOP ("outsourcing Emp sal Detail");
        SOP ("_____________________");
        SOP ("Emp Number: " + oempno + " Name " + oename);
        SOP ("No. of days worked =" + nodw);
        SOP ("wages Per day = " + wages);
```

```
    sop ("  ____________________");
    sop ("  Tatal sal = " + tsal);
    sop ("  ____________________");
  }
} // outsource Emp - BDC

class payslips
{
  psvm (String [] args)
  {
    perEmp po = new perEmp();
    po.getperEmpDet();
    po.getcompDet();
    po.displaycompDet();
    po.dispperEmp payslip();   outSource Emp op = new OutsourceEmp();
    po.getoutEmpDet();
    po.getcompDet();
    po.dispcomDet();
    po.dispoutEmppayslip();
  }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SUPER KEYWORD

Importance :-

whenever we inherit the base class features into derived class there is possibility that base class features are similar to derived class features and JVM gets an ambiguity.
In order to differentiate betn the base class features & derived class features, the base class features must be preceded by super keyword.

Syntax :-

```
super. baseclass features name
```

→ super keyword is playing an important role in three places. They are —

1. At data member level
2. At method level
3. At constructor level

1. **super at data member level :-**

→ whenever we inherit the base class data members into derived class, there is possibility that base class data member are similar to derived class data member and JVM gets a ambiguity.

→ In order to differentiate betn Base class data member & derived class data member, In the context of derived class the Base class data member must be preceded by super keyword otherwise they will be considered as current class data member.

→ ~~Syntex~~

```
Super. baseclass datamember name
```

Q. WAP which listed the concept of super keyword at data member level.

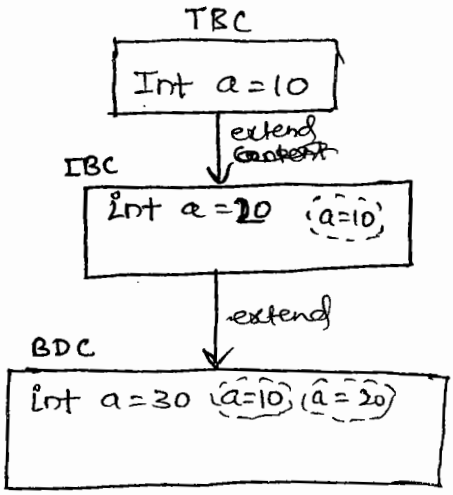
```
// SuperDemo1.java
class BC
{
  int a;
} //BC
```

```
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 from BC =" + super.a);
  sop (" val of a from DC = " + this.a);
  sop ("sum =" + c);
 }
} // DC
class supDemo1
{
 psvm (String [] args)
 {
  DC do1 = new DC ();
  do1.set (10, 20);
  do1.add ();
  do1.disp ();
 }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. How do you refer the common data members in the context of Bottom most derived class where it contain IBC, TBC?
V.V.I.

In the context of BDC we can refer current class "a" value (this.a) and IBC "a" value (super.a) but we can't refer "a" value of (super.super.a is not applicable)
The best solution to access TBC Data member (a) in the context of BDC we need to hold TBC "a" value or TBC data member (a) in temparary variable in the context of IBC & we use that IBC data member Temparary variable context access w



2. Super at method level :-

whenever we inherit the BC methods in the Context of DC there is Possibility that BC & DC methods are same and JVM gets ambiguity.

In order to differentiate bet^n BC methods & DC methods in the context of DC method the BC methods must be preceded by super Keyword otherwise the derive class method called by itself. Infinitally and gives recursive result.

This type.

Syntax:

super. base class method name

Q. WAP which listed the concept of super at method level?

```
// SuperDemo02.java
class Bc
{
 void disp()
 {
  system.out.println("BC - disp()");
```

```
    }
  } // BC
  class DC extends BC
  {
    void disp()
    {
      sop("DC-disp()");
      super.disp(); // will call class disp() from the context.
    }                   derived class disp()
} // DC
  class SupDemo2
  {
  PSVM (String [] args)
  {
    DC do1 = new DC();
    do1.disp();
  }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Super at method overridding level :—

we know that method overridding = method heading same
+
method body different

OR

The process of redefining the original method in the context of various derived classes by inheriting from base class for performing different operation is known as method overridding.

To apply method overridding concept we must apply inheritence principle.

Q. write a java Prog. which ~~listed~~ illustrate the concept of superkeyword at method overridden level.

```
// SupDemo2.java
class Circle
{
 void draw()
 {
  SOP("drawing circle - original method");
 }
} // Circle
Class Rect extentend Circle
{
 void draw()
 { super.draw(); // will call super class draw()
  SOP("drawing - Rect - overridden method");
 }
} // Rect
class SupDemo2
{
 P.S.V.M (String [] args)
 {
  Rect r = new Rect();
     r.draw();
```

Q. write a java prog which illustrate concept of super at method overridden level in multilevel inheritance.

```
// SupDemo4.java
import java.util.Scanner;
Class TBC
{
 void op(int a, int b)
 {
  System.out.println("op() - TBC - original method");
  int c = a+b;
  SOP("sum in TBC =" +c);
 }
} // TBC
```

```
class IBC extends TBC
{
  void op(int a, int b)
  {
   super.op(a,b);// will call op(-,-) of TBC from op(-,-) of IBC
   sop("op()-IBC overridden method");
   int c= a-b;
    sop("sub in IBC="+c);
  }
} // IBC
class BDC extends IBC
  {
   void op(int x1, int x2)
    {
     super.op(x1, x2);
     sop("op()-BDC overridden method");
     int m = x1 * x2;
     sop("mul in BDC="+m);
    }
} // BDC

  Class SupDemo4
  {
    Public Static Void main (string [] args)
     {
     BDC io = new BDC()
       io.op(10, 20);
     } // main
  } // class.
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

1. Q. How do you call a Base class (-) from the context of derived class (-,-) when the Both methods ( ) are same ?

Ans:- $\boxed{\text{Super. Baseclass method Name}}$

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

2. Q. How do you call a original method from the context of overridden method ( ).

$\boxed{\text{Super. original method Name}}$

The above statement must be written in the context of overridden method.

The original method resides in Base class & overridden method resides in derived class.

3. Super at Constructor level :-

1. - whenever we developed any inheritance appl^n it is highly recommended to create an object of Bottom most derive class (BDC) Because it inherit the features of TBC & IBC

2. - when we create an obj of BDC first we get the memory space for data member for TBC, second we get memory space for data members of IBC & last we get memory space for data member of BDC.

3 - In whichever order memory space is created in the same order, values will be initialize i.e. Datamembers of TBC to be initialize first, IBC (datamembers) second & Last data members of BDC w.r.t. corres-ponding classes related constructor.

- The above three points makes to understand in inheritance based appl^n constructor are calling from Bottom to Top & execution is from Top to Bottom.

- To established a comm. betn Base class constructor to Derive class constructor. we have two implicit function They are -

(a) super ( )

(b) super (...)

(a) **super()** -

It is used for calling default constructor of Base class from derive class constructor.

(b) **super(...)** -

It is used for calling Parameter constructor of Base class from derive class constructor.

☆ **Rule** —

1. when we use super ( ) or super(...) in the derive class constructors then they must be used as first executable statement otherwise we get compile time error. (Because before initialize the data member of DC, we must initialize the DM of BC.

Possibility of using super( ) and super(...)

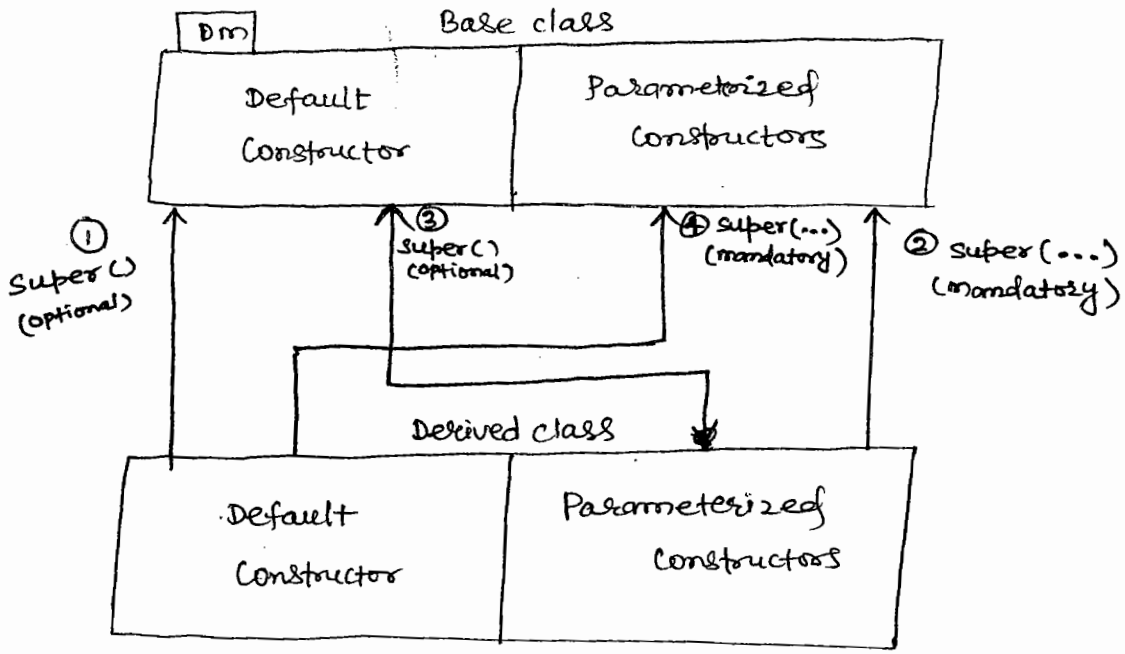


fig-①

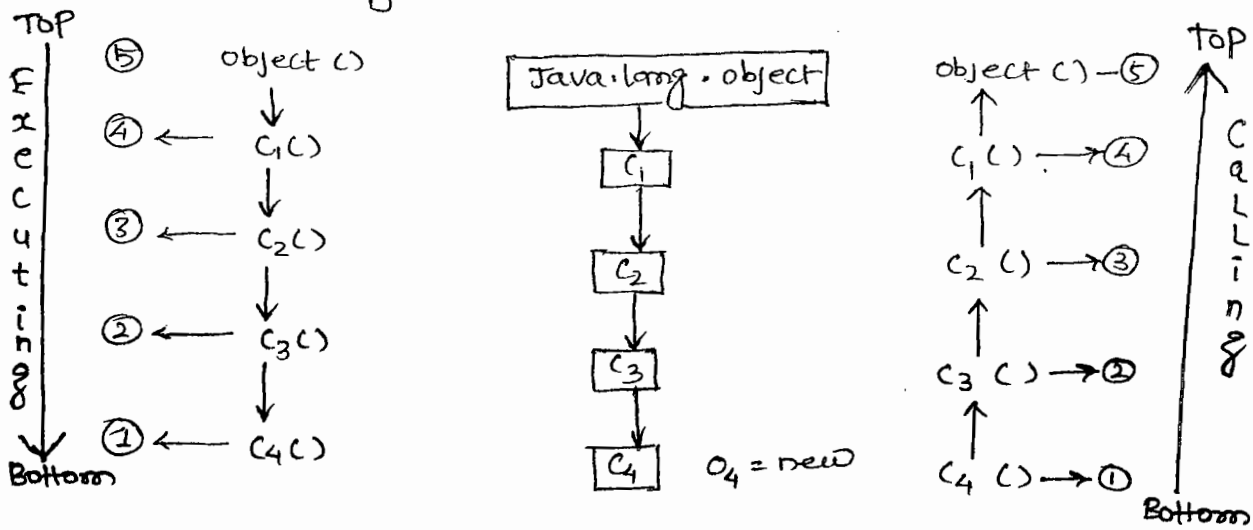
Rule (1) & (3) -

whenever derived class constructor want to call default constructor of a base class, in the context of derived class constructors we use super() and it is optional to use (Because there exist single form of Parameterized constructors)

Rule ② & ④ -

whenever derived class constructors want to call base class parameterized constructors, in the context of derived class constructor we use super(...) & it is mandatory to use (Because there is exist multiple form of Parameterized constructors)

Lets consider the following diagram which gives view about how constructors are calling & how constructors are executing



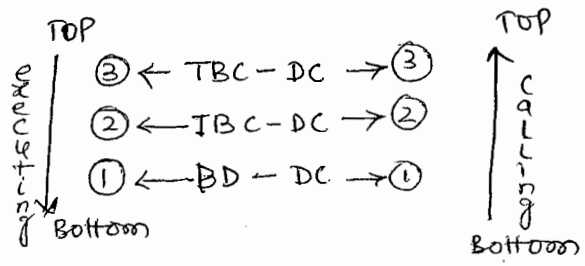
Q. write a Java prog. which illustrate the context of constructor are calling from Bottom to top and executing top to Bottom?

```
// SuperDemo5.Java
Class TBC
{
 TBC // ——— ③
 {
  // It automatically calls default constructor of
                                  Java.lang.object
  System.Out.println("TBC – DC");
 }
}// TBC
```

```
Class IBC extends TBC
 {
  IBC // ———————————— (2)
  {
   Super (); // control goes to (3)
   SOP ("IBC - DC");
  }
 } // IBC
 Class DC extends IBC
  {
   DC () // ———————— (1)
   {
    Super(); // control goes to (2)
    SOP ("Derived --DC");
   }
  } // DC
  Class SuperDemo5
  {
   P.S.V.m (String [] args)
   {
    DC dO1 = new DC (); // control go to (1)
   }
  }
```

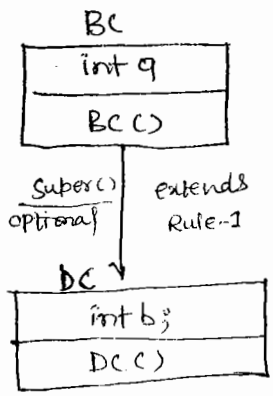
SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

<u>output</u> –



Q. write a java Program which illustrate rule (I) of fig (1)

```
//SupRule1.java
Class BC
{
 int a;
 BC() // ———————— (2)
{
 SOP( "base – DC");
 a=10;
 SOP ("value of a from BC =" +a);
}
} // BC
```



```
Class DC extends BC
{
  int b;
  DC() // ———————— ①
  {
   super(); // Rule-1 optional // control goes to ②
    SOP ("Derrived - DC");
    b=20;
    SOP ("val of b from DC" + b);
    }
  } // DC
   class SupRule1
   {
    P.S.V.m (String [] args)
    {
     DC do1 = new DC(); // control goes to ①
    }
   }
```

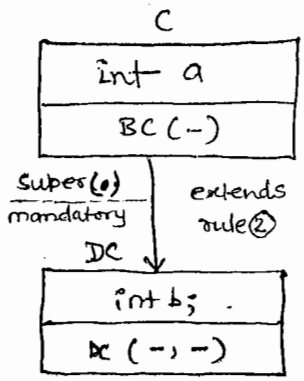
SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

O/P :-

```
base - - DC
val of a from BC=10
Derrived - -DC
val of b from DC=20
```

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

```
// SupRule 2.java
class BC
{
 int a;
  BC (int a) // ———————— ②
 {
   SOP ("base - BC");
   this.a = a;
   SOP( "val of a from BC =" + this.a);
   }
 } // BC
```



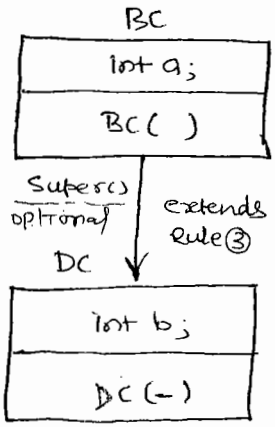
```
class DC extends BC
{
  int b;
  DC (int a, int b) // ———————— ①
  {
   super(a); // Rule-2 mandatory —— control goes to ②
   Sop ("Derrived-PC");
  }
} // DC
class SupRule2
   {
    P.s.v.m (string [] args)
    {
     DC d01 = new DC(100, 200) // control goes to ①
    }
   }
```

output –

```
base — - PC
Value of a from BC =100
Derrived ---PC
value of a from DC = 200
```

Q. write a Java Program which illustrate rule ③ of fig-①

```
// SupRule3-Java
 class BC
  {
    int a;
    BC() // ———————— ②
   }
   SOP ("base – DC");
   a=1;
   SOP (" val of a from BC=" +a);
  }
 } // BC
 class DC extends BC
  {
   int b;
   DC (int b) // ——— ①
```



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

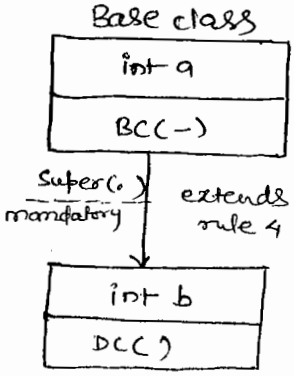
```
        super();
        SOP("Derrived—DC");
        b=x;
        System.out.println("value of b from DC="+b);
    }
}
class SupRule3
{
    public static void main(String args[])
    {
        DC d1 = new DC(7);
    }
}
```

Output -

```
base--DC
value of a from BC=1
derived--DC
val of b from DC=7
```

Q: write a java program which illustrate Rule-4, fig ①

```
// SupRule4.java
class BC
{
    int a;
    BC(int x) // ————
{
    SOP("base class--BC");
    a=x;
    SOP("val of a from BC="+a);
    }
}
class DC extends BC
{
    int b;
    DC()
    {
        super(10);
        SOP("Derrived class--DC");
        b=20;
        SOP("The val of b from DC="+b);
    }
}
```



```
class SupRule4
{
 Public Static void main (String [] args)
  {
    DC dO1 = new DC();
  }
}
```

output :

```
Base class -- pc
Val of a from BC = 10
Derrived class -- DC
Val of b from DC = 20
```

Q. What will be the output of following program ?

```
// Q.Java
class BC
{
 static
 {
 System.out.println ("BC - static Block");
 }
  BC()
  {
   SOP ("BC - DC");
  }
  {
   SOP ("BC - Instance Block");
  }
} // BC
  class DC extends BC
  {
   static
   {
    SOP ("DC - static Block");
   }
   {
    SOP ("DC - Instance Block");
   }
```

output –

```
Q - S.Block
BC -- S-Block
DC -- S-Block
BC -- Instance Block
BC -- DC
DC -- Instance Block
DC -- DC
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
    DC()
    {
      SOP ("DC-DC");
    }
  } // DC
  class Q
  {
    static
    {
      SOP ("Q-Static Block");
    }
    p.s.v.m.(String args[])
    {
      new DC();
    }
  } // Q
```

✱ this() and this(...)

In order to establish to communication betn current class Constructor we use two implicit functions. They are —

a.) this()

b.) this(...)

In otherwords the above two implicit fxn are used for establishing the communication between current class Constructor.

a.) <u>this()</u> :—

It is used for calling current class default constructor from current class parameterized constructor.

b.) <u>this(...)</u> :—

It is used for calling current class parameterized constructor from other category of constructor of same class.

Rules :-

Rule-1 - When we use this() or this(...) in the current class constructor, they must be used as first executable statement (otherwise we get compile time error).

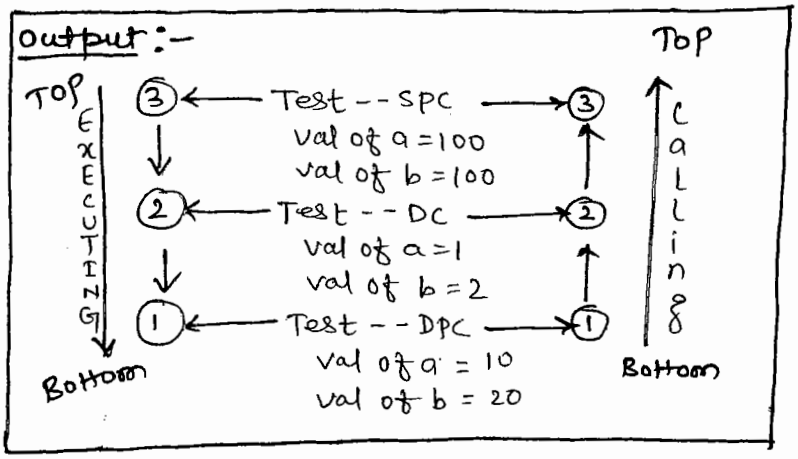
Rule-2 :- When we use this() or this(...) in the current class constructor, it is highly recommended to form only path but not recommended to form cyclic ; because cycle gives recursive constructor invocation problem.

Q: Write a Java program which illustrate this() and this(...)

```
// ThisDemo.java
Class Test
{
 int a, b;
 Test () // ———————— (2)
 {
  this(100); // control goes to (3)
  System.out.println ("Test - DC");
  a = 1;
  b = 2;
  SOP ("val of a =" +a);
  SOP ("val of b =" +b);
 }
 Test (int x) // ———————— (3)
 {
  SOP ("Test - SPC");
  a = b = x;
  SOP ("val of a =" +a);
  SOP ("val of b =" +b);
 }
 Test (int a, int b) // ———————— (1)
 {
  this(); // control goes to (2)
  SOP ("Test - DPC");
  this.a = a;
  this.b = b;
  SOP ("val of a =" + this.a);
  SOP ("val of b =" + this.b);
 }
} Test - BLC
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
class ThisDemo
{
  p.s.v.m. (String args[])
  {
    Test t1 = new Test(10, 20); // control goes to (1)
  }
}
```



Q. What is the difference between Super & this ?

'this' is always pointing to current class object and it is always used for differentiating between DM of current class & Formal parameter, Data member of current class must be preceded by 'this'.

'Super' is a keyword is always pointing to super class features & it is used to differentiate between base class features & derive class. Base class features must be preceded by 'Super' keyword.

Q. What is the difference between this() & this(...) ?

this(), this(...) are used for establishing the comm. between current class constructor.

- Super(), Super(...) are used for establishing the comm. betn Base class & Derived class constructor.

Polymorphism

Polymorphism is one of the distinct principle of OOPL and whose aim to build Scalable Application.

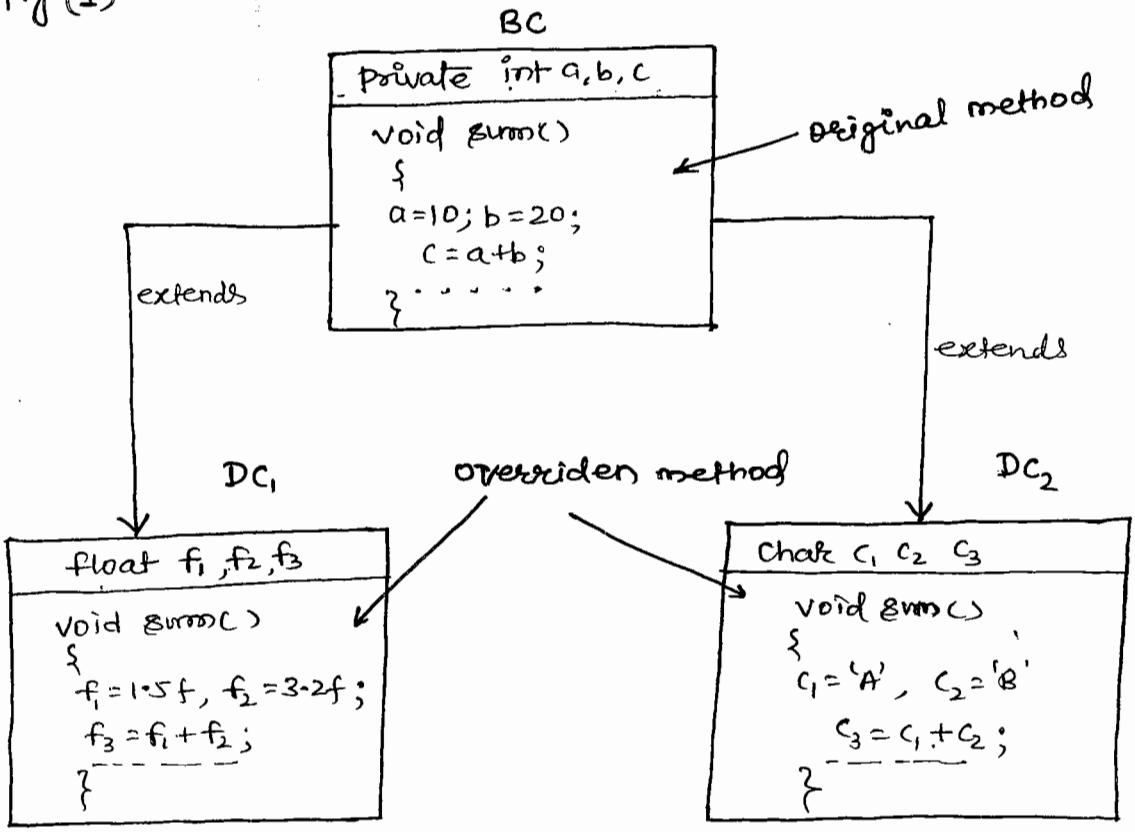
Def:- The process of representing "One form in multiple forms".

- In the definition of polymorphism one form represents original method and multiple forms represents overriden methods.
- Definition of forms
  - The state of existance of a methods; is called form.
- If a method exist in Base class it is called original form & if a method exist in Derived classes it is called overriden form.
- Polymorphism is one of the principle but not a Programming Concepts. The best principle polymorphism implemented by all the object oriented programming Languages.
- In Java programming polymorphism principle is implemented by using two Concepts. They are –
  - a.) method overriding
  - b.) method overloading
- In real world application development Polymorphism principle is always used for developing Business Logic.
- we have two types of polymorphism. They are –
  - (i) static / compile-Time Polymorphism
  - (ii) dynamic / Run-Time polymorphism
- In Java Programming we follow only dynamic polymorphism, and static polymorphism is not supported by polymorphism due to its limitations.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Let us consider the following diagram —

fig-(1)



In the above diagram sum() is actually available in one form and it is further implemented in multiple form.

Hence sum() is one of the polymorphism ().

- In the above diagram we develop the business logic in the form of dependent Business logic classes by following polymorphism principle along with method overriding.

Dynamic Binding

1.)— It is also one of the distinct principle of OOP's.

2.)— Dynamic Binding says —
"don't create the object of derived class & always create an object of Base class".

3.)— Dynamic Binding Principle is always used for executing Polymorphic based application i.e. execution logic purpose.

4.)— The Adv. of polymorphism along with method overriding and dynamic binding are —

- Appl^n takes less memory space.
- Appl^n Execution time less
- Appl^n performance is inhanced.

Def. of dynamic binding —

5.)— "The process of binding appropiriate versions of the derived class (overriden method) which are inherited from base class with base class object is called dynamic binding."

6.)— when we create an object indirectly with dynamic binding principle JVM considers two points internally. They are —

(a) what type of object ?

(b) what type of refrence object contains ?

Note :-

- To create an object indirectly with dynamic binding principle there must exist Super-sub / Base-Derived relationship. Otherwise it is not possible.

ex- `Faculty f = new Student();`

Here faculty is the super class of Student class

- Here f is an object of type - "Faculty" & f contains refrence of type - "Student".

* - Type of object is always make us to understand checking the existence of method in base-class & refrence of object makes us to understand looks for execution of the method which is overriden in the context of derived class.

- let us ~~cond~~ consider set of executable statement; which are executing the program which are represented in fig-1 (see previous page)

① BC bo1 = new BC();

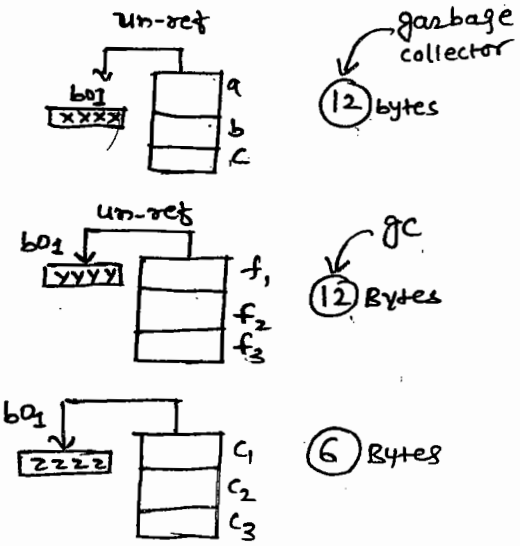
② bo1 .sum();

③ bo1 = new DC1();

④ bo1 . sum();

⑤ bo1 = new DC2();

⑥ bo1 . sum();



- In the above example –
- In Line no:s ①,③,⑤ the object bo1 contains the refrence of BC, DC1 & DC2 and hence bo1 object is known as polymorphic object.

- The method sum(); is actually available in one form and it is further implemented in multiple forms; hence sum() is known as polymorphic method.

- In line no ②, ④, ⑥ bo1 .sum() is actually only one statement but it gives multiple sum() and hence this statement is called polymorphic statement.

- In realworld applications one can solve any problem by using three cases –

Case-1 - we develop the Business logic in the form of independent Business logic class & we develop the execution logic by following direct object creation (known as plain-old execution logic (POEL)).

Case-2 :- we develop the Business logic in the form of dependent logic classes by following Polymorphism Principle along with method overriding & we develop the execution logic by following by direct object creation.

- Because of these two cases we get three limitation,
(a) Application takes more memory space.

(b) Application takes more execution time.
(c) Application performance is digraded.

case-III -

we develop the Business logic in the form of dependent Business logic classes by Following polymorphism & method overriding and we develop the execution logic by Following dynamic binding Principle.

- If we develop any application with the above case-III then such appl will get the following advantages —
  (a) Appl^n takes less less memory-space.
  (b) Appl^n takes less execution time.
  (c) Appl^n Performance is enhanced.

Q. Define Static polymorphism & dynamic polymorphism; what are the adv & limitation?

- In static polymorphism the overloaded ( ) are binding with an object at compile time; by differentiating there signatures.
- The limitation of static Prolymorphism is Poor utilization of resources. (main memory space)
- In dynamic Polymorphism overloaded methods binding with an object at run-time by differentiating with there signtuage.
- The adv. of dynamic polymorphism is effective utili-zation of resources.
- C++ programming language support both static & dynamic polymorphism but in real time C++ programmer follow only dynamic polymorphism because of its advantages.
- function overloading concept of CPP binds in there object during compile time leads to poor utiliz-ation of resources ; where as in Java Prog. overloaded method binds with object at run-time leads to effective utilization of resources.

NOTE-I :-

When we are Implementing Polymorphism with method overloading then overloaded ( ) are binding with an object at Run-time by differentiating there signature's.

NOTE-II :-

When we are Implementing polymorphism with method overriding then overriden method binds with base class object at runtime by differentiating reference of derived class present in the object of base-class.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Abstract classes in Java

we know that E & E Java prog. must starts with the concept of class i.e. without class concept there is no Java program.

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

1. Concrete class
2. Abstract class

1. Concrete class –

- A concrete class is one which contains fully defined method ment for performing specific operation.
- Defined methods of class are also known as Concrete methods or Implemented methods.
- once the class is concrete we can instancetiate (create) an obj. directly.

ex-
```
class Test
{
 void show()
 {
   sop("Test --- show()");
 }
 void disp()
  {
   sop("Test --- disp()");
  }
}
```

Here the class Test contains two defined method; hence it is called concrete class and whose object can be create directly.

```
Test t1 = new Test(); // Direct object creation.
t1.show();
t1.disp();
```

- In real world applications concrete classes are always ment for dealing with specific requirement by containing specific features and it is more suitable for Individual programmer's but not ment for dealing with common requirements.

If we choose concrete classes to deal with common requirements then such type of application gets the following limitation.

1. - Application takes more memory space (in the case of independent Concrete Business Logic classes) and also takes less memory space in the case of dependent concrete Business logic classes by following Polymorphism & method overriding.

2. - Application takes more execution time (Because we are overr-iding defined method).

3. Application performance is degraded.

To overcome the above limitation of concrete classes we use another Concept called "Abstract classes".

- In Real world Abstract classes are always ment for dealing with common requirement and they are containing common requirement and they are containing common features they are suitable for all the Programmers.

→ Definition of Abstract Class :-

- An abstract class is a coll^n of some defined methods and some un-defined methods.
- un-defined methods of a class are also known as abstract methods or un-implemented methods.

→ Definition of Abstract method :-

- An abstract methods contains only method Heading/ method declaration/ method prototype But they not contains method Body/ method definition.
- To make undefined method as abstract method the declaration of method must be preceded by abstract keyword otherwise we get compile time error.

Syntax –

abstract ReturnType MethodName (List of formal Parameter Types if any);

ex –

```
abstract void sum ();
abstract void op (int, int);
abstract void draw ();
```

- We know that E & E method belongs to class even abstract method.
- If any class of java contains abstract method then it is called abstract class and to make the class abstract then whose definition proceed by abstract keyword otherwise we get compile-time error.

- syntax for abstract class –

```
abstract class <clsname>
  {
    --------
    --------
abstract Ret.Type MethodName ( List of Formal Parameter Types, if any );
    --------
    --------
  }
```

ex:-
```
abstract class operation
  {
    --------
    --------
    abstract void sum ();
    abstract void op (int, int);
    abstract void draw ();
  }
```

Here, operation class is one of the abstract class & whose obj can't be instancetiate directly but it can be intancetiated indirectly.

- The Reason for not creating an obj. of abstract class directly is that there is no use of calling abstract method.

ex – Operation O1 = new Operation (); // Invalid

Q. Write a Java Program which illustrate the concept of abstract classes (compute sum of two integer, two float).

```
// SumIntFloat.Java
Import Java.util.Scanner;
abstract class operations
  {
    Scanner S = new Scanner (System.in);
    abstract void sum ();
  }
```

```
class Isum extends operations
{
 int a, b, c;
 void sum()
 {
  sop ("Enter the first Int value:");
    a = Integer.parseInt(s.nextline());
  sop ("Enter the Second Int value:");
    b = Integer.parseInt(s.nextLine());
     c = a+b;
   sop ("Int sum in Isum =" +c);
 }
} // Isum - sub/Implementation class

 class Fsum extends operations
  {
   float a, b, c;
   void sum()
   {
    sop ("Enter the first Float value:");
    a = Float.parseFloat(s.nextLine());
    sop ("Enter the Second Float value:");
    b = Float.parseFloat(s.nextline());
     c = a+b;
     sop ("Float sum in Fsum=" +c);
   }
 } // Fsum - sub/Implementation class
}
}

   class . SumIntFloat
   {
    psvm (String [] args)
  { // operations o1 = new operations();    Invalid
    operation o1 = new Isum();
          o1.sum();
          o1 = new Fsum();
           o1.sum();
  } //main
} // class - ELC
```

- when the object memory space is collected by garbage collector

The following are the reasons, where the g.c. collects unrefrenced memory space

- when the object is re-refrencing to some other class object, previous object memory space becomes un-refrence collected by g.c. and adds to Heap memory.

- when the current class refrenced objects made explicitly as NULL and hence current class obj. memory space becomes unrefrenced and collected by garbage collector & adds to Heap memory.

Q. write a Java program which will compute area of different figures such as rectangle, square, circle ?

```
// AreaCRS.java
import java.util.*;
abstract class AreaDemo
{
  float area;
  Scanner s = new Scanner (System.in);
  abstract void area ();
}

class circle extends AreaDemo
{
  int r;
  void area ()
  {
   SOP ("enter the value of r");
   int r = Integer.parseInt (s.nextLine());
   area = (3.14f * r * r);
   SOP (" ____________");
   SOP ("area of the circle =" + area);
   SOP (" ____________");
  }
} // class circle
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
class rectangle extends AreaDemo
{
  int l, b;
  Void area()
  {
   System.out.println("enter the value of l:");
   int l = Integer.parseInt(s.nextLine());
   sop("enter the value of b:");
   int b = Integer.parseInt(s.nextline());
   area = l*b;
   sop("_____");
   sop("Area of rectangle=" +area);
   sop("_____");
  }
}
class square extends AreaDemo
{ int f;
  Void area()
  {
   sop("enter the value of f:");
   int f = Integer.parseInt(s.nextLine());
   int area = f*f;
   sop("_____");
   sop("Area of Square =" +area);
   sop("_____");
  }
}
  Class AreaCRS
  {
   P S V m (String [] args)
   {
    AreaDemo a = new circle();
      a.area();
      a = new rectangle();
       a.area();
      a= new square();
       a.area();
   }
  }
```

Q. write a java program which will compute perimeter of diff. shape like circle, square, Rectangle.

```
// PeriDemo1.java
import java.util.*;
abstract class PeriDemo
{
  float Peri;
  Scanner s = new Scanner (System.in);
  abstract void Peri();
}
class circle extends PeriDemo
{
  int r;
  void Peri()
  {
   sop ("enter the value of radius r:");
   int r = Integer.parseInt (s.nextLine());
   Peri = (3.14f * 2 * r);
   sop ("_____");
   sop ("Perimeter of circle =" +Peri);
   sop ("_____");
  } // peri()
} // class circle
class Rectangle extends PeriDemo
{ int l, b;
  void peri()
  {
   sop ("enter the value of length l:");
   int l = Integer.parseInt (s.nextLine());
     sop ("enter the value of width b:");
    int b = Integer.parseInt (s.nextLine());
    Peri = (2* (l+b));
   sop ("_____");
   sop ("Perimeter of rectangle =" +Peri);
   sop ("_____");
  }
}
```

```
class square extends PeriDemo
{
  int f;
  void peri()
  {
    SOP ("enter the value of side f:");
    int f = Integer.parseInt(s.nextLine());
    int peri = 4*f*f;
    SOP ("_________________");
    SOP ("perimeter of square:"+peri);
    SOP ("_________________");
  }
}

  class PeriDemoI
  {
    PSVM (String [] args)
    {
      PeriDemo p = new Circle()
        p.peri();
        p = new rectangle()
          p.peri();
        p = new square()
          p.peri();
    }
  }
```

* Abstract Base class & Abstract Derived class :-

Def. of Abstract base class (ABC):-

- ABC class contain physical representation of abstract method which are always re-usable by implementation classes.

Def. of Abstract derived class

- ADC class contains Logical representation of abstract methods which are inherited from abstract base class.

- ABC & ADC class are always re-usable by implementation classes w.r.t. both ABC & ADC classes we can't instancetiate there obj. directly but we can create an object directly w.r.t. Concrete implementation classes.

* - The following or the reason to make the derived class as Abstract derived class —

(a) If the derived class inherits multiple abstract methods & if is not defined atleast one method then the current derived class is known as abstract and whose definition must be made as an abstract by using abstract Keyword otherwise we get compile time error.

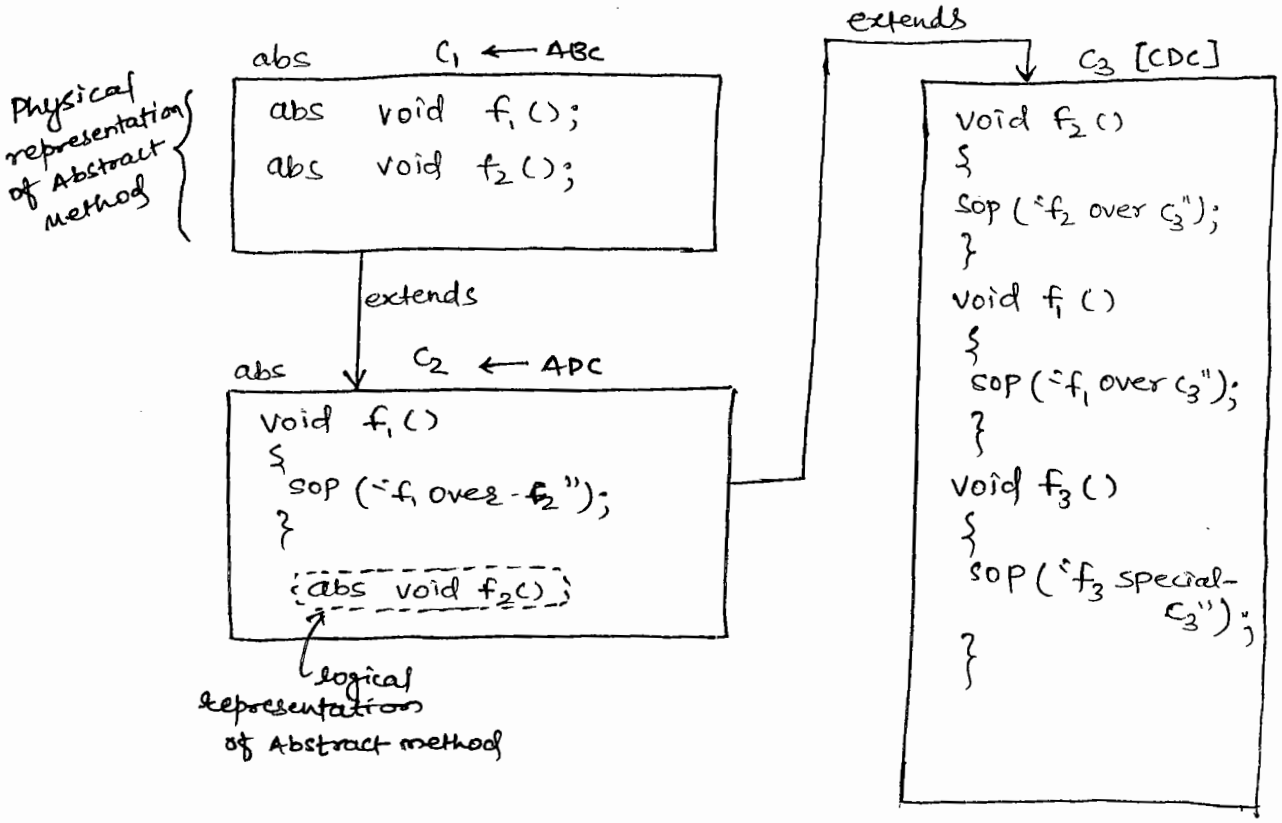
(b) If the derived class contains full of defined method & if the derived class programmer added physical representation of abstract method then the current derived class also called abstract derived class and whose definition must be made as abstract by using Abstract keyword otherwise we get compile time Error.

Q. write a Java Prog. which illustrate the concept of abstract base class & abstract derived class.

OR

Implement the following diagram (see next Page) —

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.



CDC ⇒ Concrete derived class

```
// AbDemo2.Java
abstract class C1
{ abstract          void f1();
  abstract  void f2();
 } // C1 - ABC
  abstract class C2 extends C1
   {  void f1()
       {
        sop ("f(1) - overriden - C2");
       }
  } // C2 - ADC
  class C3 extends C2
     { void f2()
       {
        sop ("f(2) - overriden - C3");
       }
```

```
void f1 ()
 {
  sop ("f1() - overriden - C3");
 }
void f3()
 {
  sop("f3() - special - C3");
 }
} // C3 - CDC
class AbDemo2
 {
  psvm (string args[])
  {
    sop ("w.r.t. C3 - CDC - Direct object creation");
    C3 O3 = new C3();
        O3.f1();
        O3.f2();
        O3.f3();
    sop ("w.r.t C2 - Indirect object creation - Dynamic Binding");
  // C2 O2 = new C2(); invalid because C2 is abstract
        C2 O2 = new C3();
          O2.f1();
          O2.f2();
        // O2.f3(); invalid; f3() is not exist in C2
    sop ("w.r.t C1 - Indirect object creation - Dynamic binding");
  // C1 O1 = new C1(); invalid
  // C1 O1 = new C2(); invalid
        C1 O1 = new C3();
          O1.f1();
          O1.f2();
      // O1.f3(); invalid
  }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Points to be Remember —

* 1. Abstract classes always contain common reusable features.
2. E & E abstract class must participates in inheritance.
3. Java Programming doesnot contain final abstract classes in other words since abstract class is participated in inheritance, whose definition should not be final.
4. E & E abstract class must contain atleast one implemention class.
5. An object of abstract class can't be initiated (created) directly but it can be create indirectly w.r.t. implementation class.

an object of abstract class = an object of implementation subclass

6. Every abstract class of java implements Polymorphism for business logic development and uses dynamic binding principle for execution logic.
7. The advantages of abstract class based Appl^n are
   a) Less memory space.
   b) Less Execution time.
   c) more performance.
8. An object of abstract class can be declared but it can't be refrenced directly by itself and it can be refrenced indirectly w.r.t. Concrete Implementation class.

```
AC ao1 ; // valid
ao1 = new AC(); // invalid
ao1 = new CC(); // valid
```

9. An abstract class of Java can contain defined main method.
10. E & E abstract method must be instance() but it Shouldn't be Static because instance method can do repeated operation (overriding) where static only one operation Performed.

11. Abstract method of Java Shouldn't be private and final.

12. Polymorphism & dynamic binding principles to be applied abstract, instance method, defined instance method & overriden instance method and not recommended to apply static defined method.

Even though we override static define method JVM will also call static original method but not static overriden method.

13. this, super, this(), this(...), super() & super(...) are used in the context of Non-static and we should not use in static context (compile-time-error)

14. programmatically it is not recommended to Java programmer to define programmer define constructor in abstract classes because direct refrencing is not possible.

15. In an abstract class one can write instance block and static block but they never participate in inheritance.

Q. What the output of the following program ?

```
// ADBDemo4.java
abstract class AC
{
 static
  {
   sop("Ac-static");
  }
  {
   sop("Ac- Instance block");
  }
 AC()
  {
   sop("Ac-DC");
  }
 abstract void f1();
} // AC
class CC extends AC
{
 static
  {
   sop("CC-static");
  }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
{
  sop ("cc-Instance block");
}
cc(int x)
{
  sop ("cc-pc");
}
void f1()
{
  sop ("f1() - overriden-cc");
}
} //cc

class ABDemo4
{
  PSVM (String [] args)
  {

  // Ac ao1 = new Ac();
      Ac ao1 = new cc(10);
        ao1.f1();
  }
}
```

output :-

```
Ac --- static
cc --- static
Ac --- Instance Block
Ac --- DC
cc --- Instance Block
cc --- PC
f1() --- overrideo        --cc
```

✱ Limitation of Abstract Class:-

1. Abstract classes provides only common reusable features and unable to provide universal common reusable features.

2. Abstract classes never participates in multiple inheritance.

3. We know that Abstract class is a collection of defined methods and abstract methods.
   Overriding defined method of Abstract class takes more execution time & overriding abstract methods of abstract method takes less execution time. Overall Abstract class based application takes partially less execution time and unable to provide completly less execution time.

To overcome these limition we must use the concept of Interfaces.

———x———x———x———

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Interfaces.

- Programmatically the concept of Interfaces makes us to develop universal user-defined data type.
- To develop universal-user-define data type with interfaces concept we use a keyword called "Interface".
- Each & Every Interface name is acting as universal user-defined data type.

Adv. of Interfaces; over Abstract classes —

- If we develop any appl^n with interfaces concept rather than abstract classes then that application get following advantages.

(a) Interfaces concept provides universal common reusable features.

(b) Interfaces concept participates in multiple Inheritance.

(c) Interfaces related Application provides Completly Less execution time.

Conclusion:-

The overall scope of interface is more than Abstract class scope.

Definition of Interfaces:-

1- An Interface is a collection of public static Final XXX data member and public abstract method.

(OR)

2. An Interface is a collection of universal common reusable data member & universal common reusable methods.

Syntax for defining Interface:-

```
Interface <intf name>
{
    variable declaration cum initialization;
    methods declaration;
}
```

- "Interface" is a keyword used for developing universal user define data type.
- <intfname> represents a Java valid variable name treated as a name of a interface. Every interface name in Java is treated as universal user define data type.
- Variable declaration represents data members of an interface. Every data member of interface is ~~use~~ ment for universally reusable and hence they must be initalize otherwise we get compile time error.
  Since the data members of interface are universally accessable, whose values can't be changed (final), whose memory space be created only once (static), ~~until~~ and they can access over the Java program (public).
  Hence, Each & Every data member of Interface is by default belongs to public static final xxx data member.
  here xxx represents data type, variable name, variable value.

  ex:-

  $$\underbrace{\text{Public static final}}_{\text{Appended by Java Compiler Env.}} \quad \underbrace{\overset{x}{\text{float}}\ \overset{x}{\text{PI}} = \overset{x}{3.1417f}}_{\text{written by programmer}};$$

- methods declaration represents list of un-defined methods ment for performing universal operation. Since the methods of interface ~~are~~ are undefine; to make this methods as abstract we need not to write abstract keyword explicitly (by default abstract) to make this method universally accessable; explicitly we need not to write Public keyword (by default public).
  Hence every method of interface is by default belongs to public abstract.

  ex:-

  $$\underbrace{\text{Public abstract}}_{\text{Appended by Java compiler env.}} \quad \underbrace{\text{void draw ();}}_{\text{written by Java Programmer.}}$$

* - An object of interface can't be instantiated directly but it can be created directly w.r.t. implementation class.

ex:- Define an Interface sathya technology by choosing suitable DM and suitable method.

```
// SathyaTechI.java
Interface SathyaTechI
  {
    int a = 10;
    int b = 20;
    void show();
    void draw();
  }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Compile the above program —

```
Javac SathyaTechI.Java
```

ensure sathyaTechI.class file will be generated.

- In general for every Interface definition we get a .class file.

view the profile of an interface —

```
JavaP SathyaTechI
```

```
O/P :- interface SathyaTechI {
        Public static final int a;
        Public static final int b;
        Public abstract void show();
        Public abstract void draw();
```

Since, the data members of interface belongs to static & they must be access w.r.t. interface name.

```
S.O.P ("val of a =" + SathyaTechI.a);
SOP ("val of b =" + SathyaTechI.b);
```

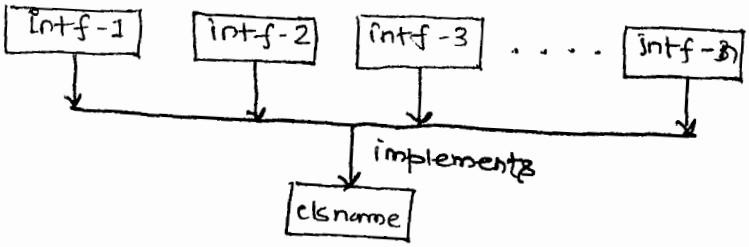
* Inheriting the features of Interfaces :-

In Java Programming we have three approaches to inherit the Features of Interfaces. They are –

* Approach-I :-

- This approach makes us to understand how to inherit the Features from base interfaces into derived class.

Diag:-

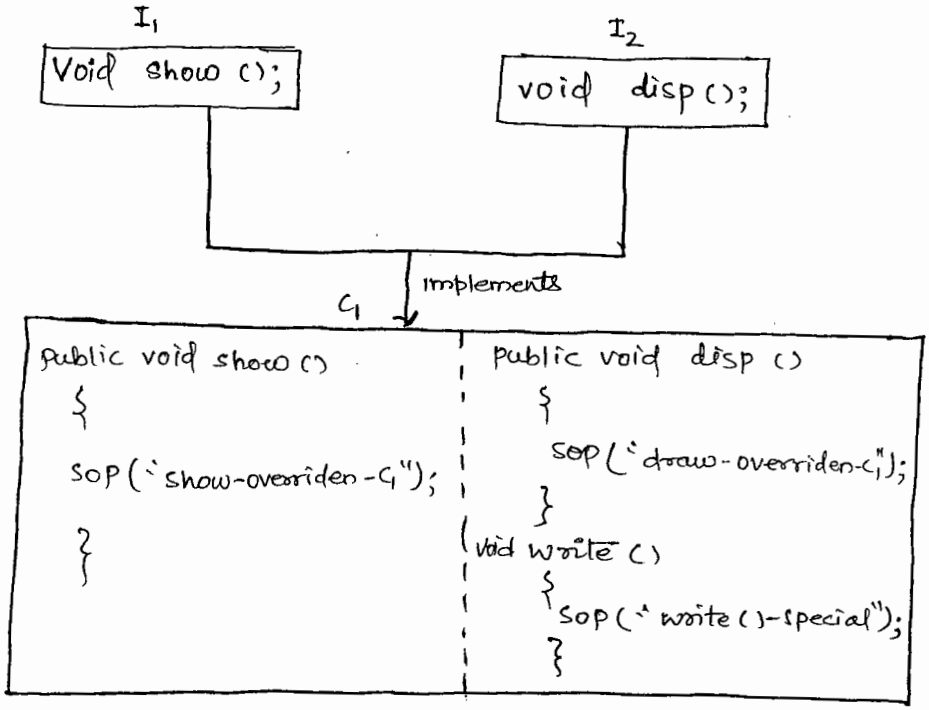


Syntax:-

```
[abstract] class <clsname> implements <intf-1><intf-2>...<intf-n>
{
    var Declaration
    method Def/Decl;
}
```

- In the above syntax <clsname> represents name of the derived class.
- <intf1>.<intf-2>...<intfn> represents name of the Base interfaces.
- implements is a KWD used for inheriting the features of Base interfaces into derived class + its provide richest functionality to derived class.
- If a derived class inherites multiple abstract methods from multiple interfaces and if the derived class is not defining atleast one abstract method then the derived class is known as abstract derived class and whose definition must be made as abstract by abstract keyword otherwise we get compile time error.
- In Java prog. 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.

Q. write a Java Program which will implement the following diagram –

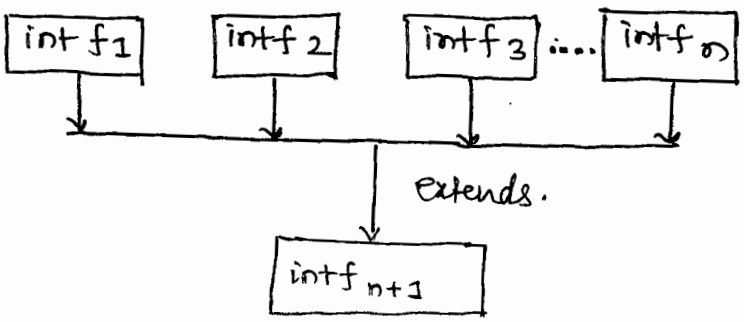



```
// IntfProg1.Java
interface I1
{
  void show();// Public abstract void show ()
}
interface I2
{
  void disp ();// Public abstract void disp ()
}
class C1 implements I1, I2
{
  Public void show ()
  {
    sop ("show () - overriden - C1");
  }
  Public void disp ()
  {
    sop ("disp () - overriden - C1");
  }
  void write ()
  {
    sop ("write () - special - C1");
  }
} // C1 - implementation/derrived class I1 & I2
```

```
class IntfProg1
{
 psvm(String [] args)
 {
  sop("w.r.t. C1 - DOC");
    C1 O1 = new C1();
      O1.show();
      O1.disp();
      O1.write();
    sop("w.r.t. I1 - IN - DOC - DB");
   // I1 io1 = new I1(); invalid becoz I1 is abstract
      I1 io1 = new C1();
      io1.show();
   /* io1.disp(); invalid bcoz disp() & write() doesn't exist
      io1.write(); */                          in I1.
    sop("w.r.t. I2 - IN - DOC - DB");
   // I2 io2 = new I2(); invalid bcoz I2 is abstract
      I2 io2 = new C1();
        io2.disp();
   /* io2.show(); invalid bcoz show() & write() doesn't exist
      io2.write(); */                          in I2.
 }
}
```

☆ Approach-2 (Interface inheritence)

- This approach makes us to understand how to inherit the base interface(s) into derived interface.



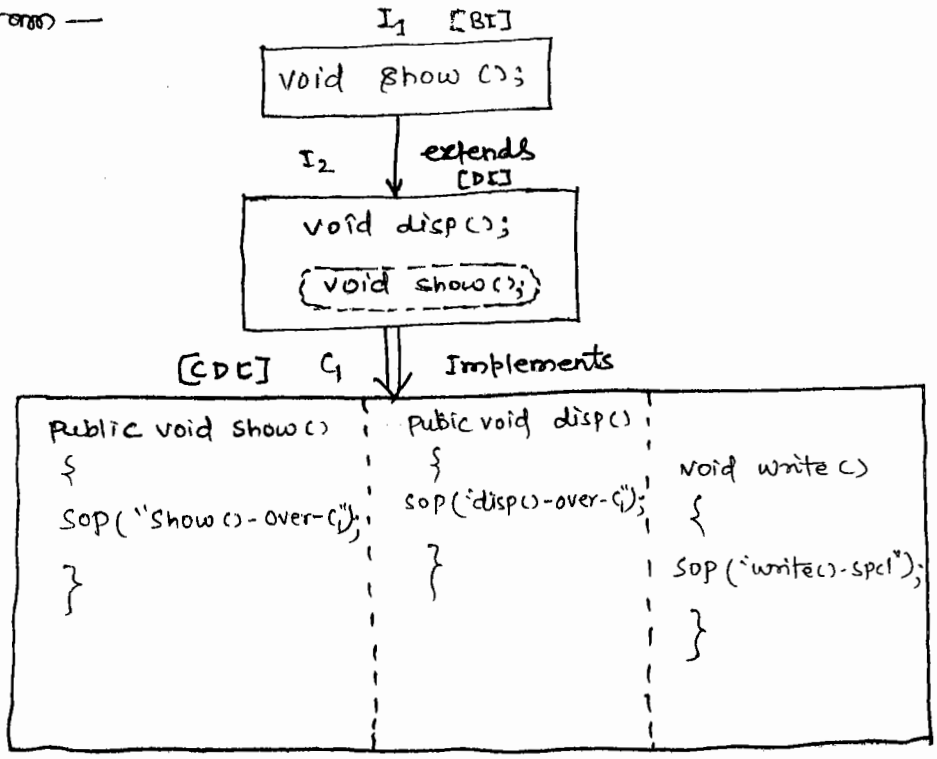
Syntax

```
interface <intf n+1> extends <intf1> <intf2> ----<intfn>
    {
      var. decl. cum initalization
      method Decl;
    }
```

- In the above syntax <intf n+1> represent name of the derived interface(s);
- In the above syntax <intf1> <intf2> --- <intfn> represents name of base interfaces.
- "extends" is a keyword used to inherit the features of base interface(s) into derive interface + provides richest functionality to derived interface.

***- In Java prog. one derived can extends only one Base class whereas one derived interface can extends either one or more than one interface.

Q. write the Java prog. which will implement the following diagram —



```
interface I1
 {
  void f1();
 }
interface I2 extends I1
  {
   void f2();
  }
class C1 implements I2
  {
    Public void f1()
    {
     sop ("f1 - override - C1");
    }
   Public void f2()
    {
     sop ("f2 - override - C1");
    }
   void f3()
    {
     sop ("f3 -- special - C1");
    }
  } // class C1 — BLC
 Class INFProg
 {
  Public static void main (string [] args)
    {
      C1 co1 = new C1();
       co1.f1();
       co1.f2();
       co1.f3();
      I1 io1 = new C1();
       io1.f1();
      I2 io2 = new C1();
       io2.f1();
       io2.f2();
    }
 } // ELC - main()
```

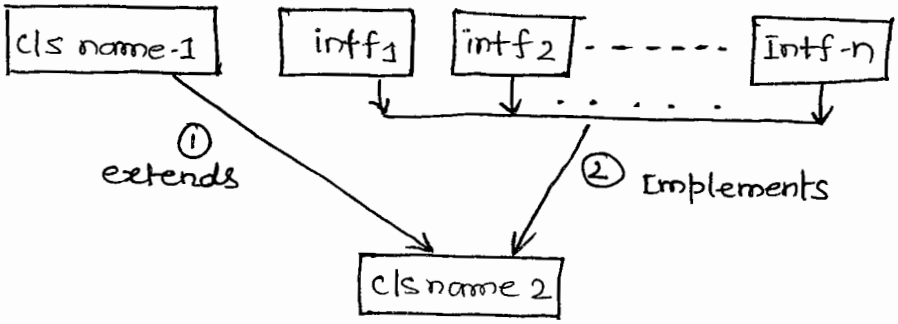
Output

```
f1 —— override — C1
f2 —— override — C1
f3 —— special — C1
f1 —— override — C1
f1 —— override — C1
f2 —— override — C1
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Approach-3 [Parallel inheritence]

This approach makes us to understand how to inherit the features parallely from Base class and from Base Interfaces into Derived class.



Syntax –

```
[abstract] Class <clsname-2> extends <cls name-1>
                                implements <intf1> <intf2> --- <intfn>
  {
     var decl;
     method Def/Decl;
  }
```

explanation:—

- In the above syntax <clsname-1>, <clsname-2> represents name of the Base & Derived classes.
- <intf1> <intf2> --- <intfn> represents Base Interfaces name.
- 'extends' & 'implements' are the two keywords used for the inheriting the features of Base class & Base Interfaces into derived class + They provides richest functionality to derived class.
- When we use Both 'extends' & 'Implements' in a single syntax then we must use 'extends' keyword first & later we must use 'implements' keyword otherwise we get compile time error.

Consequences of Abstract Classes :-

1. If any CC of Java containing purely Null body Methods then there is no use of calling Null body methods w.r.t corresponding class object because Null body methods never gives any result. Hence, it is highly recommended to make the corresponding CC difinition as abstract by the abstract keyword.

Definition of Null body method :-

- A Null body method is one of the defined method with no block of statement.

Syntax

```
ret-type methodName (list of formal Params)
  {
    // null body
  }
```

- most of the null body methods present in Abstract classes. Every Null body method of abstract class must be overridden.

Q. write a java program which illustrate the concept of Null body methods.

```
// AbstractDemo5.Java
abstract class parent
 {
  void goc() // null body method
  }
} // Parent - ACC
class child extends parent
 {
  void goc ()
   {
    SOP ("going to college e & e day");
   }
 } // child - impl. class
class AbstractDemo5
 {
  PSVM (String [] args)
   {
    Parent P = new child();
 } }   P.goc();
```

2. Industry highly recommended to override abstract methods of abstract classes / Interfaces and not recommended to override defined methods of concrete classes because method overhead cost of overriding ~~is~~ abstract method is less than method overhead cost of overriding defined methods.

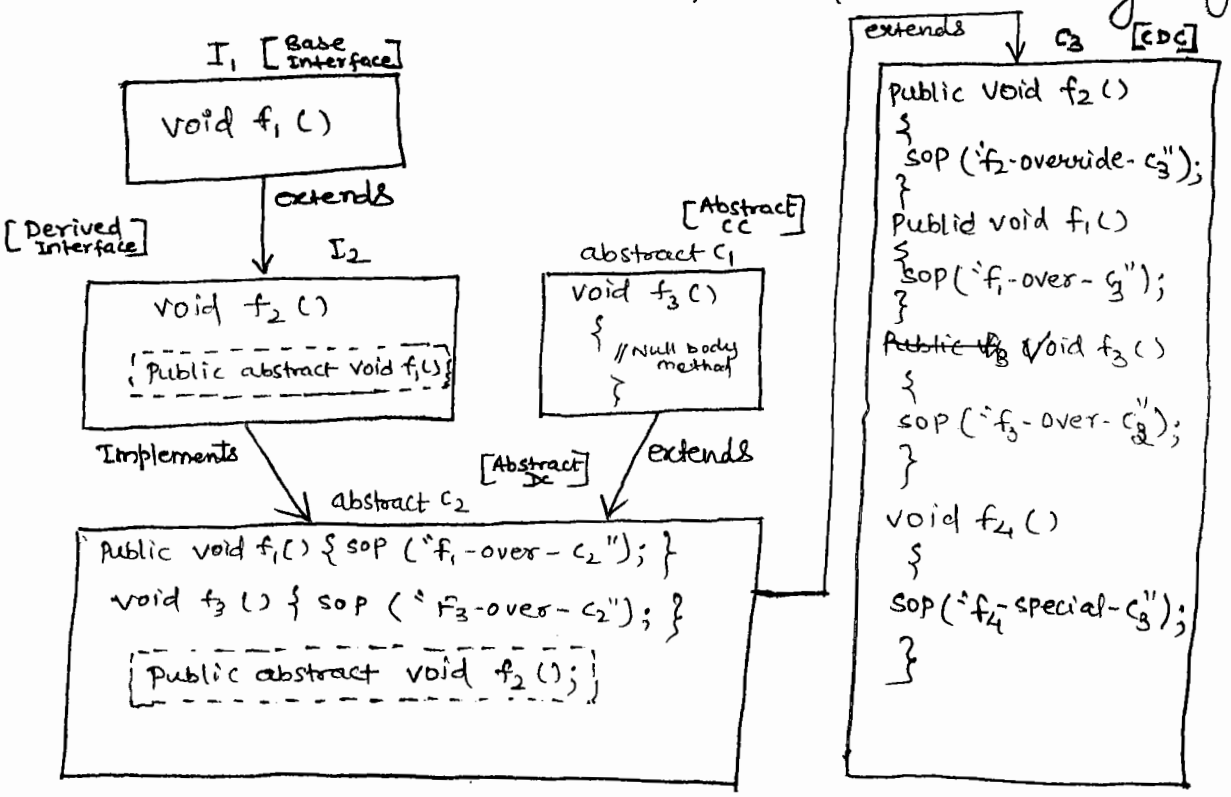
(Not recomm.) method overhead cost for overriding def. method } = { Cost required for eliminating Existing Body + Cost required for constructing new Body

(recommended) method overhead cost for overriding Abstract method = Cost req. for constructing new Body only.

- more method overhead cost takes more execution time and less overhead cost takes less execution time.

Q. Can we make a concrete class as abstract ?
Yes,
reason - See consequence (1.).

Q. Write a Java program which illustrat the concept of approach-3 of interfaces or implements the following diag :-



```
// IntfProg2.java
interface I1
{
 void f1();  // public abstract void f1()
} // I1 - base interface
interface I2 extends I1
 {
  void f2();  // public abstract void f2()
 } // I2 - Derived interface
 abstract class C1
  {
   void f3()  // null body method
    {

    }
  } // C1 - ACC
  abstract class C2 extends C1 implements I2
   {
    Public void f1()
     {
      sop ("f1() - overriden - C2");
     }
  void f3()
    {
     sop ("f3() - overriden - C2");
    }
  } // C2 - ADC
    class C3 extends C2
     {
      public void f2()
       {
        sop ("f2() - overriden - C3");
       }
      public void f1()
       {sop ("f1 - overriden - C3");
       }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
void f3 ()
  {
    sop ("f3 () - overriden - C3");
  }
void f4 ()
  {
    sop ("f4 () - special - C3");
  }
}// C3 - CDC
class IntfProg2
{
  psvm (String [] args)
  {
    sop ("w.r.t. C3 - DOC");
    C3 O3 = new C3 ();
      O3.f1 ();
      O3.f2 ();
      O3.f3 ();
      O3.f4 ();
    sop ("w.r.t. C2 - IN DOC - DB");
    // C2 O2 = new C2 ();  invalid bcoz C2 is abstract.
      C2 O2 = new C3 ();
        O2.f1 ();
        O2.f2 ();
        O2.f3 ();
    //  O2.f4 ();  invalid bcoz f4() doesn't exists in C2
    sop ("w.r.t. C1 - IN-DOC - DB");
    // C1 O1 = new C1 ();  invalid bcoz C1 is abstract
    // C1 O1 = new C2 ();  invalid bcoz C2 is also abstract
      C1 O1 = new C3 ();
        O1.f3 ();
    /* O1.f1 ();  invalid f1(), f2(), & f4() doesn't exist in C1
       O1.f2 ();
       O1.f4 ();  */
```

```
SOP ("w.r.t. I2 - IN-DOC-DB");
// I2 io2 = new I2 (); invalid bcoz I2 is abstract
// I2 io2 = new C2 (); invalid bcoz C2 is also abstract
   I2 io2 = new C3 ();
      io2 . f1 ();
      io2 . f2 ();
/*    io2 . f3 ();   invalid f3() & f4() doesnot exist in I2
      io2 . f4 ();  */
SOP ("w.r.t. I1 - IN-DOC-DB");
/* I1 io1 = new I1 ();    invalid bcoz I1, I2 & C2 are abstract
   I1 io1 = new I2 ();
   I1 io1 = new C2 ();  */
   I1 io1 = new C3 ();
      io1 . f1 ();
/* io1 . f2 ();  invalid f2 (), f3 () & f4 () doesn't exist in I1
   io1 . f3 ();
   io1 . f4 ();  */
}
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

✱ INNER / NESTED INTERFACES

- Let us assume there exist a normal interface with collection of abstract method. If any class implements this normal interface which is containing multiple Abstract.

  abstract methods then the derived class will becomes Abstract.

  w.r.t. to abstract class we can't create object directly and unable to call intrusted overriden method.

- In order to call intrusted overriden method the derived class programmer may override un-intrusted abstract methods also to make the derived class as Concrete. But this process is one of the time consuming process and not recommended in industries.

- To overcome this time consuming process & to define only intrusted method by leaving un-intrusted method we use the concept of nested/Inner interfaces.

Advantages of Inner/Nested Interfaces:-

- Allows us to override only required abstract method by eliminating un-intrusted abstract method.
- Appl° development time is less.
- The overall performance of inner interfaces is much better than normal interfaces.

Definition of Inner/Nested Interfaces:-

- If we define an interface 'Y' within the definition of another interface 'X'. Then interface 'Y' is called Inner/Nested interface and interface 'X' is called outer interface.

ex:-

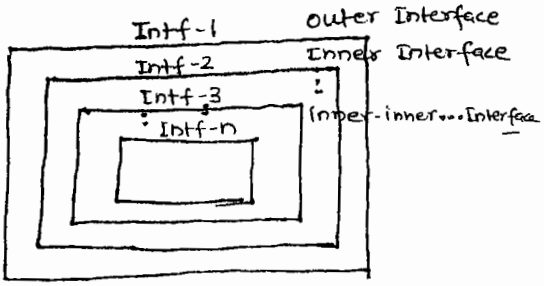
```
interface X      <---- outer interface
{
  -------
  interface Y    <---- inner interface
  {
    -------
  }
}
```

Definition of Containership

The process of defining one interface definition in another interface definition is called containership.

Syntax 1:-

```
interface <intf-1>          outer interface
{
  ------
  ------
  interface <intf-2>        inner interface
  {
    ------
    ------
    :
    interface <intf-n>      inner-inner...inner interface.
    {
      ------
      ------
    }
  }
}
```



Example:-

```
interface x
{
  interface y
  {
    interface z
    {
      ------
      ------
    }
  }
}
```

When we compile the above program we get the following .class file —

x.class
x$y.class
xyz.class

} resides in secondary memory

Syntax for accessing :-

<outer Interface> . <inner Interface> . <inner Inner Interface> . . . . . .

ex:- Access inner Intf 'Y'

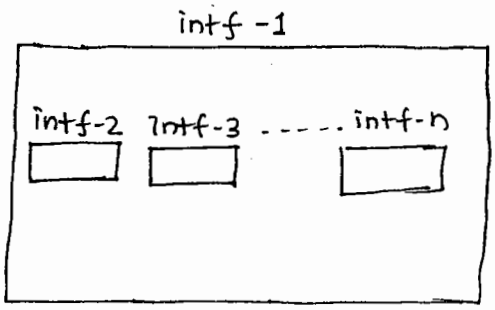
x.Y

Access inner Intf 'Z'

x.y.Z

Syntax-2 :-

```
interface <intf-1>
{
  interface <intf-2>
  {
   ____
   ____
  }
  interface <intf-3>
  {
   ____
   ____
  }
  interface <intf-n>
  {
   ____
   ____
  }
}
```



Skelton ex:-

```
interface x
{
  interface Y
  {
   }
  interface Z
  {
   }
}
```

- If we compile the above program then we get the following .class file :-

x.class
x$Y.class
x$Z.class

Q. Develop anytype of arithmatic operation for any type of number using inner interface?

```
// InnerIntfDemo.java
import java.util.Scanner;
interface AOp
{
  interface Sum
  {
    void add ();
  } // sum- inner intf
  interface Sub
  {
    void Sub ();
  } // Sub-inner intf
  interface Mul
  {
    void mul ();
  } // mul-inner intf
} // AOp

class SumOp implements AOp.Sum
{
  int a, b, c;
  public void add ()
  {
    Scanner S = new Scanner(System.in);
    sop(" Enter the first value:");
    a= Integer.parseInt (s.nextLine());
    sop(" Enter the Second value:");
    b= Integer.parseInt (s.nextline());
    c=a+b;
    sop ("Sum="+c);
  }
} // SumOp
class MulOp implements AOp.Mul
  {
    int a,b, c;
    public void mul ()
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
Scanner s = new Scanner(System.in);
   sop ("enter the first val");
  a = Integer.parseInt( s.nextLine());
   sop ("enter the second val");
   b = Integer.parseInt( s.nextLine());
    c = a*b;
    sop ("mul="+c);
  }
} // mul-op

class InnerIntfDemo
  {
   psvm (String [] args)
    {
      sop ("w.r.t SumOp - DOC");
      SumOp so = new SumOp();
         so.add();
      sop ("w.r.t. AOp.Sum - In-DOC-DB");
   // AOp.Sum as = new AOp.Sum();  invalid bcoz AOp.Sum
                                   is Abstract.
      AOp.Sum as = new SumOp();
           as.add();
```

Functional Interface:-

- Definition of functional interface:--
- A functional interface is one which contain only one abstract method.
- Functional interface is the new facility is available from JDK 1.8 version onwards.
- In Java Programming every abstract method of interface can be overridden either in the form of explicit implementation class or in a form of anonymous inner class.

Q. write a Java program which illustrate the concept of functional interface & override the abstract method w.r.t anonymous Inner class.

```
// AINC.Java
interface India.
{
 void makeIndia();
} // india - functional interface.
Class AISC
{
 PSVM (String [] args)
 {
  india io1 = new India() ———————— (i)
  {
   Public void makeIndia()
   {
    sop ("Java Prof. Participating in making in india");
   }
  };
  IO1 . makeIndia();
  India io2 = new India() ———————— (ii)
  {
   Public void makeIndia()
   {
    sop ("H/w Prof. participating in making in india");
   }
  };
```

```
io2.makeIndia ();
new India () // nameless object ———————(iii)
{
 Public void makeIndia
 {
  SOP ("xyz prof. Participating in making in india");
 }

}.makeIndia ();
} // main
} // AINC - ELC
```

→ when the above program is compiled for (i), (ii) & (iii) implicitly we get three anonymous inner classes and they are labeled as respectively:—

AINC$1
AINC$2
AINC$3

→ E & E anonymous inner class is implicitly final so, that we can't extend it.

→ For E & E anonymous inner class there exist an implicit super class called java.lang.object.

→ For E & E anonymous inner class there exist super interface which is represented according to our specification in the program.

Q. write a Java Program which illustrate the concept of an interface contains multiple abstract method and they overridden in the form of anonymous inner classes and call those methods w.r.t. nameless object.

```
// AINCEX.Java
interface SathyaTech
{
 void write();
 SathyaTech show();
 SathyaTech disp();
}
```

```
Class AINCEX
{
  P.S.V.M (String [] args)
  {
    new SathyaTech() // Nameless object
    {
      public SathyaTech show()
      {
        SOP ("show() - overridden - AINCEX $ 1");
        return this; // here 'this' is refers an obj. of anyno-
                                                        mous
                        inner class, which is nothing but an obj of
                        SathyaTech interface
      }
      public SathyaTech disp()
      {
        SOP ("disp() - overridden - AINCEX $ 1");
        return this;
      }
      public void write()
      {
        SOP ("write() - overridden - AINCEX $ 1");
      }
    }.disp().show().write();
  } // main
} // AINCEX
```

Q. write a Java Program which will compute sum of two integers, two float & two Strings with inner interfaces along with anynomous inner interfaces.

```
// AINCEX2.Java
import Java.util.Scanner;
interface AOP1
{
  interface Sum1
  {
    void add();
  } // sum - inner functional Inf
} // AOP1 - outer inf.
```

```
class AINCEX2
{
 PSVM (string [] args)
 {
  new AOP1.Sum1() // ———————(i) —— AINCEX2$1
  {
   int a, b;
   Public void add()
   {.
    Scanner s = new Scanner (System.in);
    sop ("enter first int val");
        a = Integer.parseInt (s.nextLine());
    sop ("enter Second int val");
     b = Integer.parseInt (s.nextLine ());
    sop ("int sum =" +(a+b));
   }

  }.add();
  new AOP1.Sum1() // ———————(ii) —— AINCEX2$2
  {
   Public void add()
   {
    Scanner s = new Scanner (System.in);
    sop ("enter first float val");
    float a = Float.parseFloat (s.nextLine());
    sop ("enter Second Float val");
    float b = Float.parseFloat (s.nextLine());
    sop ("Float Sum=" +(a+b));
   }

  }.add();
  new AOP1.Sum1() // ———————(iii) —— AINCEX2$3
  {  String a, b;
   Public void add()
   {
    Scanner s = new Scanner (System.in);
```

```
sop ("enter first string val");
a = s.nextline();
sop ("enter second string val");
b = s.nextline();
sop ("string concat =" +a+b);
}
}.add();
}
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

PACKAGES

- In general to learn any programming language, we must learn its Concepts, Syntaxes & Library.
- To learn 'c' language, we need to learn its concepts, syntaxes and its library.
- 'c' language Library is a collection of header files, A header file is a collection of predefine common functions In 'c' software development if any function is common then such common function must be placed in header file, Hence header files always contains common functions.

Similarly Learning Java is nothing but learning its concepts, OOPS concept & its API (Application Programming Interface)

Definition of API :-

An API is a coll^n of pre-define Packages.

- Definition of Package :-
  - A Package is a collection of classes, interfaces and sub-Packages. A sub Packages interms contains collection of class, interfaces & sub-sub Packages, etc.

**- The purpose of Package is that to place common classes and common Interfaces. another words If a class & Interface are common for most of Java programmers then such common classes & Interfaces must be placed in packages.

Advantages of Package :-

- If we develop any Java Appl^n with the concept of Packages than we get following advantages :-

1.) Appl^n development time is Less.
2.) Appl^n memory space is Less.
3.) Appl^n execution time is Less.
4.) Appl^n Performance is an enhanced.
5.) Redundancy of the code is minimized.
6.) Able to get the slogan of Java (WORA).
7.) Provides uniqueness among duplicate classes or eliminate

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- ambiguity problem between duplicate classes & interfaces.

- Types of Packages:-

We have three types of Packages they are -

1.- Predefine/ Build-In Packages
2.- User/Programmer/custom-Define Packages
3.- Third-Party Packages.

1. Predefine-Package -

This packages are developed by SUN-MicroSystem Developers available as a part of Java software and whose role is to deal with universal requirements.

2. Userdefine Package:-

User-define packages are developed by Java Programmer available as a part of Java Project & whose role is to deal with common requirement.

3. Third-Party Package:

Third party packages are developed by third-Party S/W vendors, available as a part of third Party S/W which are used by Java programmers to communicate with third-Party software.

ex:-

① oracle.jdbc.driver.OracleDriver

② ibm.jdbc.driver.DB2Driver

✱ Types of predefine Packages:-

We have three types of pre-define Packages. They are -

1. Core (J2SE) Packages:
   These Packages are used to developed client side Appl.
2. Adv (J2EE) Packages:-
   These Packages are used to developed server-side Appl.
3. Mobile/Micro (J2ME) Packages:-
   Used to developed Mobile/ wireless Appl.

- **List of predefine Packages in J2SE**

In J2SE we have 8-essential Packages ; they are —

1. Java.Lang.*

It is one of the predefine Package and it provide Language facility / functionality.

Some of the Lang. functionalities are —

(a) Displaying the data on the consol (System).
(b) Accepting the command line argument (String).
(c) obtaining the garbage collection facilities (object).
(d) Data Conversions (wrapper classes)
(e) Development of exception based Application (Exception, Error)
(f) Development of thread based Appl^n (Thread, Runnable).

This is the Package which are imported by default to E & E Java Program ; and hence this Package is called Default Package.

2. Java.awt.* (Abstract Windowing Toolkit)

- The purpose of this Package is to design GUI Applications (Look & Feel Appl^n)
- To design GUI Application, we need GUI components or controls, such as Label, Buttons, Textfield, Panel, etc.
- In Java Programming all GUI components are available in the form of pre-define classes such as Label, Button, Checkbox, Textfield, Panel etc.
- Creating a GUI component is nothing but creating an object of appropriate pre-define class.

Ex:- Create a Button component with a name SAVE.

```
Button b = new Button("SAVE");
```

[SAVE]

3. Java.awt.event.*

Here event is the sub-Package of awt Package.
Event-Package contains Collection of predefine classes, interfaces and they are used for obtaining functionality or behaviour of GUI application.

In other word Java.awt.event.* is used for providing the functionality to GUI application.

Hence,

To develop complete GUI Application, we need to import Java.awt.* and Java.awt.event.*

4.) Java.io.* (File Programming)

- The purpose of this Package is to get Data Persistency.
- As a Part of J2SE we can achieve the data Persistency by using files (By using Database s/w also).

→ Definition of Stream -

The flow of data in the form of bytes or bits between main memory & secondary memory is known as Stream.

→ Definition of Input Stream -

Transferring the data from file of secondary memory or Keyboard into main memory is called input Stream.

→ Definition of Output Stream -

Transfer the temporary data of main memory into the file of secondary memory or on the console (Monitor).

5.) Java.applet.* (applet programming)

- The purpose of this Package is to develop distributed application.
- In the initial days of Java development, SUN MICROSYSTEM developers develop a concept called Applets. whose aim to developed distributed application with the help of pre-define class called Applet.
- Java.applet.* contains only one predefine class Applet and whose fully qualified name is Java.applet.Applet.
- Definition of applet -

An applet is Java Prog. runs in the context of browser/ WWW and whose results are shatable across the universe.

6.) Java.net.* (Network Programming)

- The Purpose of this Package is to develop client-server Appl^n to Share the data between multiple machine.
- In client server Appl^n development we need to develop two types of Programs —

(a) client side Program (b) server side program.

- To develop client-side program & server side program we have predefine classes & interfaces which are present in Java.net.*;

7.) Java.util.*; (Collection FrameWork)

- The Purpose of this Package is to develop high performance based Appl^n
- Definition of C.F.W.
  It is the one of the additional service or standard Mechanism developed by SUN-MICROSYSTEM developers and whose role is grouping multiple values either of same type or different type or all the types in a single variable with dynamic size. This single variable is called CFW variable.

8.) Java.text.* (Text Processing)

The Purposes of this Package are —

ⓐ Formating of Dates
ⓑ Formating of Times
ⓒ Manupulation of Numerical-values etc.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Development of User/Programmer Define Package

- User Define Packages are developed by Java Programmer, available as a Part of Java Project and whose role is to deal with common requirements.

- The purpose of user-define package creation is that to Place common classes and interfaces in the package.

- what are all the guidelines are followed by SUN-MICROSYSTEM developers for developing Pre-define Packages, same guidelines will be followed by Java Programmer for development of User-define Packages.

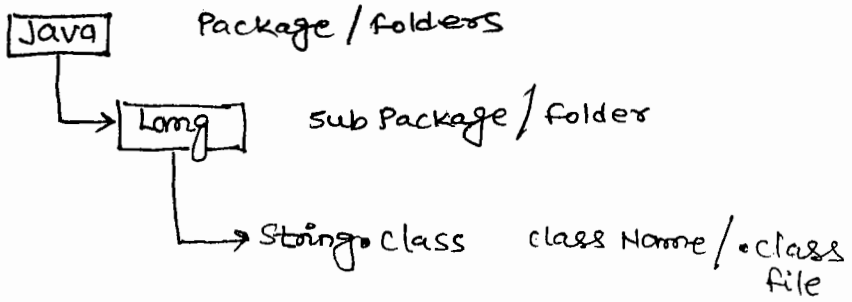
> "creating user define package is nothing but creating it as a directory/folder in secondary memory in current working Machine".

Consider the following statement.

Java.Long.String

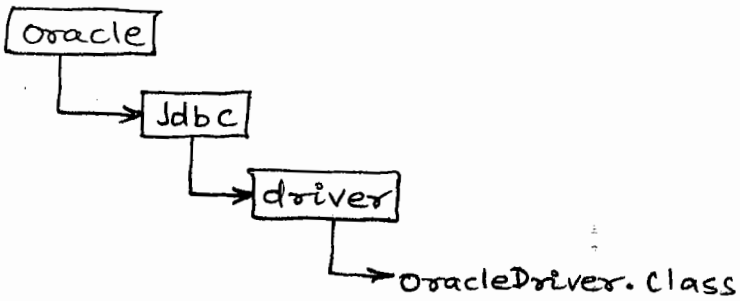
Java point of view	O.S. point of view
'Java' - outer Package	'Java' - Root Folder
'Long' - sub Package	'Long' - Sub Folder
'String' - class Name	'String' - .class file

Package structure for above statement :-



Q. Draw the Package diagram for the following statement:-

Oracle.Jdbc.driver.OracleDriver



NO's of Phases required for dealing with User-define Packages:-

In order to deal with user-define Packages we have four Phases-

1. create a Package.
2. Place the Classes & interfaces in the package.
3. Compile the classes & interfaces of the package.
4. Use the classes & interfaces of the package.

Phase-I (Creating User-define Package)

To create a user-define Package we need to use Following Syntax -

$$\text{Package} \quad Pack_1[\cdot Pack_2[\cdot\cdot\cdot[\cdot Pack_n]]];$$

Explanation -

- Package is a keyword used to developed user define Package.
- $Pack_1, Pack_2 \cdots Pack_n$ represents Java valid variable name treated as user define Package.
- $Pack_1, Pack_3 \cdots Pack_n$ represents sub-Package specification is optional because in Package may or mayn't contain subpackage.
- $Pack_1$ represents - outer package specification is mandatory.

ex:- package P_1;
or
package $P_1 \cdot P_2$;
} Package Statement

Rule:- Package Statement must be first executable Stat; outherwise compile time error.

Phase-2 (Steps/Guidelines for placing classes & interfaces in Package)

- In order to place the common classes & interfaces in the Package we must follow the following guidelines:-

1. - choose an appropriate Package Name for placing common classes and Interfaces.

2. choose an appropriate ClassName / InterFace Name and insure that whose modifier is must be public.

3. The modifier of the constructor of the class which is present in the package must be public. (This rule is not applicable in the case of interfaces, because Interfaces doesn't contain constructors)

4. The modifier of the method of the class/Interface which is present in the package must be public (This rule is optional in the case of interfaces, because every interfaces method belong to public)

5. whatever the ClassName or InterFace Name Present in the package must be given as FileName with an extension .Java.

6. At any point of time in one window, either we need to define class definition or interface def. within the package but not both the definition because we get File recognization problem. (multiple public classes/interfaces are not permitted).

Ex- Create a Package tp and place a class called Test with Suitable constructors & Methods

```
//Test.Java   //————————(5)
Package tp; //————————(1)
Public class Test //————————(2)
{
  Public Test() //————————(3)
  {
    sop ("Test-Dc");
  }
  Public void disp () //————————(4)
  { sop ("Test - disp()");
}  }
```

Q. create the Package tp and place an interface itest

```
// Itest.Java // ———————— (5)
Package tp; // ———————— (1)
public interface Itest // ———— (2)
{
                    // ———— (3) - not applicable
  void show();      // ———— (4)
}
```

Phase-3 (compiling the classes and interfaces of the Package)

- In order to compile the classes and interfaces of the package we use the following syntax.

Javac -d . Filename.Java

Explanation -

1. Here -d is an option/switch which gives indication to Java Env. saying that go to FileName.Java, take the PackageName and automatically it will be created as a directory/Folder provided the following condition must be satisfied :-
   (i) FileName.Java should not contain any errors.
   (ii) Earlier the PackageName shouldn't be created as a directory. (if it is already created as directory then it will be carry forwarded without recreating it). and it will be considered as current directory.

   Hence -d is called directory option.

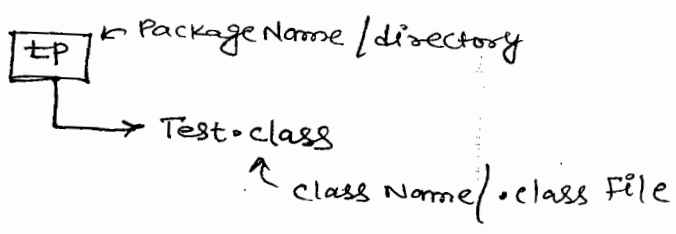
2. Javac is the tool and it will compile FileName.Java and generates FileName.class.

3. 'dot (.) refers current directory. due to dot (.) currently generated .class file will be automatically copied into currently created directory i.e. PackageName.

ex.1 - Compile Test.Java (See Previous Pages

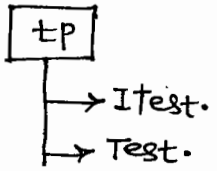
Javac -d . Test.Java

Package diagram



ex-2 compile Itest.Java (see Previous Pages)

```
Javac -d . Itest.Java
```



Phase-4 (No. of approaches to use the classes & Interfaces of the Package)

- In order to use the classes and interfaces of the Package, we have two approaches -
  - (a) By using import Statement. Approach
  - (b) By using full Qualified Name Approach.

a.) By using import Statement :-

"import" is a Keyword which is used for referring either all the classes and interfaces of the package or a particular class or particular infaces of a specific package.

we can use import Statement in Five ways they are -

approach-I

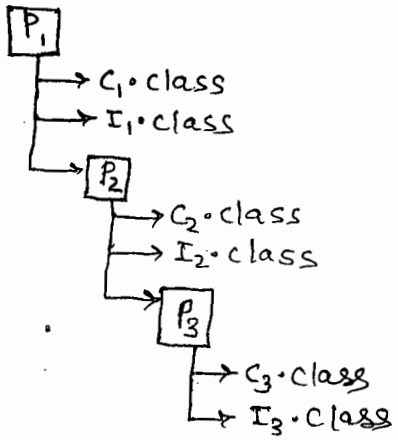
Syntax: import $Pack_1[.Pack_2[....[Pack_n]]].*$;

This approach makes us to understand how to refer all the classes and interfaces of a Particular Package.

Here "*" represents a wild character which gives or provides Complete classes & interface definitions Present in the particular Package.

ex:-

1. import $P_1.*$;
2. import $P_1.P_2.*$;
3. import $P_1.P_2.P_3.*$;



- When we use Stat. ① as a part of Java Program then we can refer all the class and interfaces of Package P_1 but not P_2 and P_3 Packages.

- When we use Statement ② as a part of our Java Program, we can refer all the classes and interfaces of package P_2 but not P_1 and P_3 Packages.

Approach-II

Syntax: import $Pack_1[.Pack_2[\ldots[.Pack_n]]]$.class/Interface Name;

This approach makes us to understand how to refer a particular class or particular interface of a specific package.

eg:-

① import $P_1.C_1$;

② import $P_1.P_2.I_2$;

③ import $P_1.P_2.P_3.C_3$;

- When we use State ① as a part of our Java Program then we can refer only class C_1 of P_1 Package, but we can't refer other classes & interfaces of P_1 Packages.

Conclusion -

Hence industry is highly recommended to use approach ②; because it provides required classes and interfaces and not recommended to use approach-① because it provides unintrested classes and interfaces along with required class & interfaces and leads to higher memory space in the main memory.

★ Static import Statement

- It is one of the new facility available from JDK 1.5 Version onwards
- Static import Statement is applicable to Static DataMembers & Static methods.
- The purpose of SIS is that eliminating redundent referring of class/Interfaces name before static data members and static methods.

  In other words static import stat. makes us to refer Static DM and static methods directly without preceded by class/Interface Name.

Static import can be use in three ways –

approach- III (Static DataMember)

Syntax:-

```
import static Pack1[.Pack2[....[.Packn]]].class/InterFace Name.
                                          Static Datamember Name;
```

approach -IV (static methods)

syntax:-

```
import static Pack1[.Pack2[....[.Packn]]].class Name.Static Method Name;
```

approach- V (All Static Features)

Syntax:-

```
import static Pack1[.Pack2[....[.Pack-n]]].class/Interfacename.*;
```

Q- WAJP which illustrate the purpose of static import statement at static data member level, Static Method level and all the static features level.

```
// SIS.Java
/* import static ip.Inst.name;
   import static ip.Inst.loc;  (OR) */
   import static ip.Inst.*;
/* import static Java.Lang.Math.PI;
   import static Java.lang.Math.E;
   import static Java.lang.Math.Pow;  (OR) */
   import static Java.lang.Math.*;
   import static Java.lang.system.out;
   class SIS
   {
     psvm (String [] args)
     {
      out.Println("w.r.t. ip.Inst");
      out.Println("_____");
      op("name of inst ="+name);
      op("Location of inst ="+loc);
      op("_____").
```

```
        op ("w.r.t. java.lang.math");
        op ("_____");
        op ("val of PI ="+PI);
        op ("val of e ="+E);
        op ("pow(4,3)="+pow(4,3));
        op ("pow(14,3)="+pow(14,3));
        op ("_____");
    }
}
```

b) <u>By using full Qualified Name Approach</u> :-

- This approach is also known as canonical Form.
- If any class/interface is completely Qualified by Package hierarchy then it is called fully Qualified Name.
- full Qualified Name is an alternative Mechanism for referring a class/interface of a Package instead of using import statement.
- If we use any class or interface, Multiple times in our Java program then it should be referred by its complete Package hierarchy.

Syntax :-

$$\boxed{Pack_1 [.Pack_2 [\cdots [.Pack_n]]] . Class/Interface\ Name}$$

1. ex :- class satya extends $P_1 . C_1$
```
{
}
```

2. ex - class Hyd implements $P_1 . P_2 . I_2$
```
{
}
```

3. ex $P_1 . P_2 . P_3 . C_3 \ . O_3 =$ new $P_1 . P_2 . P_3 \ C_3()$.

Q. WAJP which Makes use of Test class of tp (see previous page)

```
// PackDemo1.java
import tp.Test;
class PackDemo1
{
 psvm (String [] args)
 {
  Test t1 = new Test();
   t1.disp();
 }
}
```

Q. WAJP which makes use of itest interface of tp Package [see previous pages.]

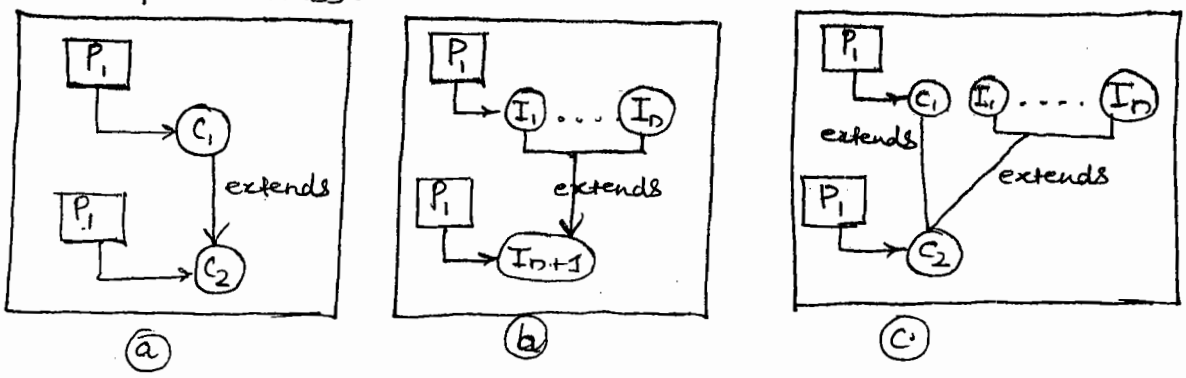
```
// PackDemo2.java
class Sathya implements tp.Itest
{
 public void show()
 {
  sop ("show() - overridden- Sathya");
 }
} // sathya - Implementation class
class PackDemo2
{
 psvm (String [] args)
 {
  sop ("w.r.t. Itest - in-dir-obj - dynamic binding");
  tp.Itest io = new Sathya();
   io.show();
  sop ("w.r.t. Itest - AINC");
   new tp.Itest()
   {
    public void Show()
    {
     sop ("show() - overridden - AINC");
    }
   }.show();
 }
}
```

Models of Application in Packages:-

In real-world application we have three types of Models in Packages they are —

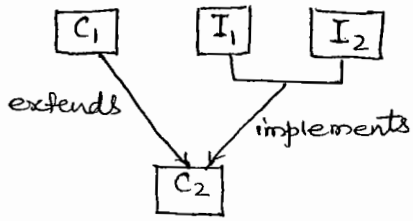
Model-I -

This model makes us to understand How to reuse the Feature within the package between class-class, Interface-Interface, Interfaces – class.



Q: WAJP which will implement the diagram (c)

Consider the following diagram

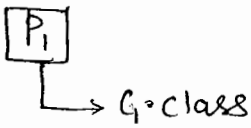


Here C_1, C_2, I_1, I_2 belongs to P_1

```
// C1.Java
Package P1;
Public abstract class C1
{
 Public void f1() // Null body
                  Method.
 }
}
```

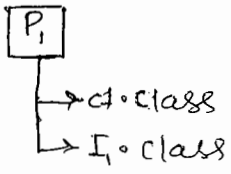
Compile the program

```
Javac -d . C1.Java
```

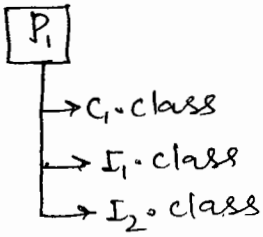


SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
// I1.java
Package P1;
Public interface I1
{
 void f2();
}
```

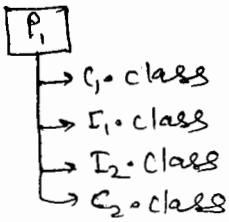



```
// I2.Java
Package P1;
Public interface I2
{
 void f3();
}
```




```
// C2.Java
Package P1;
Public class C2 extends C1 implements I1, I2
{
  Public void f1()
  {
    SOP("f1() - overridden - C2");
  }
  Public void f2()
  {
    SOP("f2() - overridden - C2");
  }
  Public void f3()
  {
    SOP("f3() - overridden - C2");
  }
} // C2 - implementation class C1; I1 & I2
```

compile above Programs

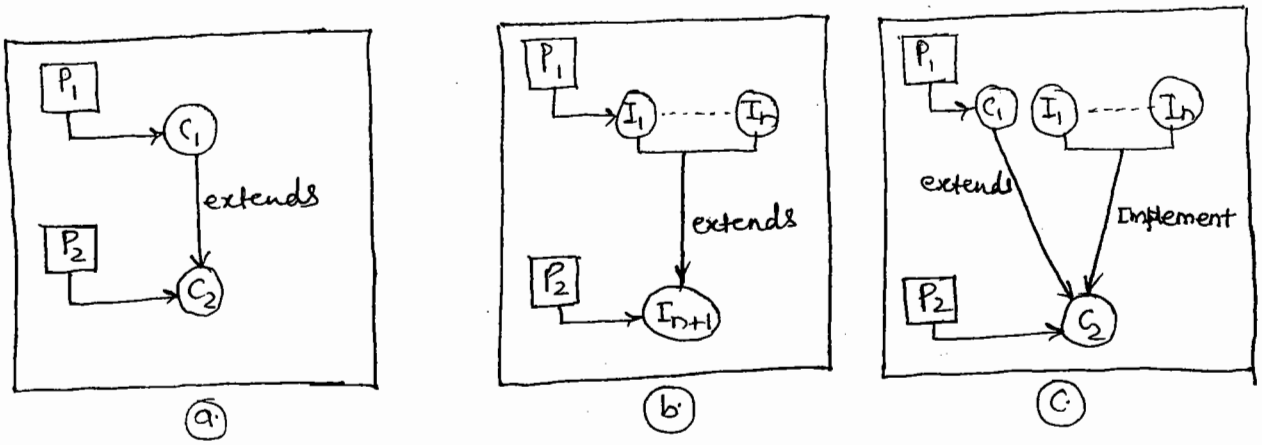


write a Java main program which makes use of C_1, C_2, I_1, I_2 of Package P_1.

```
// PackDemo3.Java
import P1.C2;
class PackDemo3
{
 PSVM (String [] args)
 {
  SOP ("w.r.t. — C2 — DOC");
   C2 O2 = new C2();
    O2.f1();
    O2.f2();
    O2.f3();
  SOP ("w.r.t - C1 - IN-DOC");
// P1.C1 O1 = new P1.C1(); invalid, bcoz C1 is abstract
   P1.C1 O1 = new C2();
    O1.f1();
/* O1.f2(); invalid; bcoz f2() & f3() does not exist in C1
   O1.f3(); */
   SOP ("w.r.t - I1 - IN-DOC");
   P1.I1 IO1 = new C2();
    IO1.f2();
/* IO1.f1(); invalid bcoz f1() & f3() doesnot exist in I1
   IO1.f3(); */
   SOP ("w.r.t — I2 - IN-DOC");
   P1.I2 IO2 = new C2();
      IO2.f3();
/*    IO1.f2(); invalid bcoz f2() & f1() doesnot exist in I2
      IO1.f1(); */
 }
```

Model-II.

This model makes us to understand How to reuse the features across the Packages between class-class, Interface-Interface, Interface-class



Q. WAJP which will implement diagram (b.)

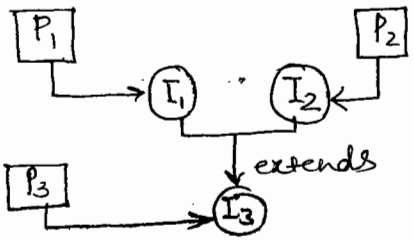
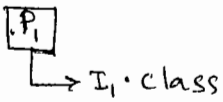


fig-(b)

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

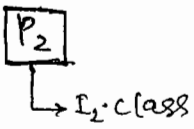

```
// I1.Java
  Package P1;
  Public interface I1
  {
    Void f1();
  }
```

compile
Javac -d . I1.Java




```
// I2.Java
  Package P2;
  Public interface I2
  {
    void f2();
  }
```

Compile
Javac -d . I2.Java



```
// I3.Java
Package P3;
Public interface I3 extends P1.I1, P2.I2
{
  void f3();
}
```

Since, all the above three interfaces containg abstract Method, to override those Method either we need to take explicit Implementation class or anonymous inner class.

The following main program follows anonymous inner classes to overridde the method of above interfaces.

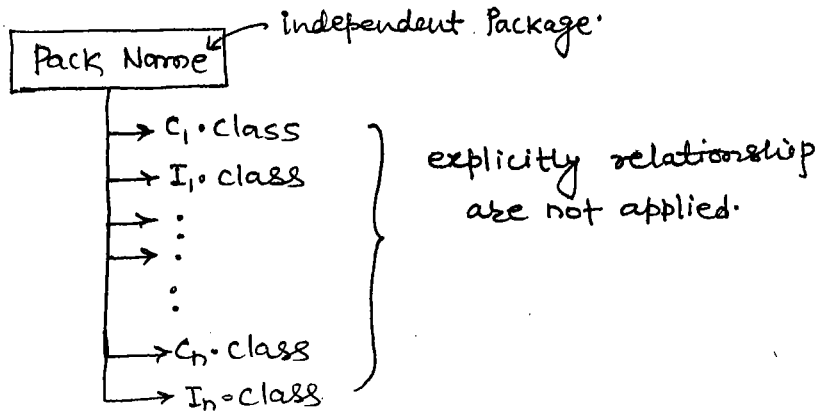
```
// PackDemo4.Java
Class PackDemo4
{
  Public static void main (String [] args)
  {
    SOP ("w.r.t. I1 - AINC");
    P1.I1 io1 = new P1.I1 () // AINC - PackDemo4 $ 1
    {
      Public void f1()
      {
        SOP("f1() - overridden - AINC");
      }
    };
    io1.f1();
    SOP ("w.r.t. I2 - AINC");
    P2.I2 io2 = new P2.I2 // AINC - PackDemo4 $ 2
    {
      Public void f2()
      {
        SOP ("f2() - overridden - AINC");
      }
    };
    io2.f2();
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

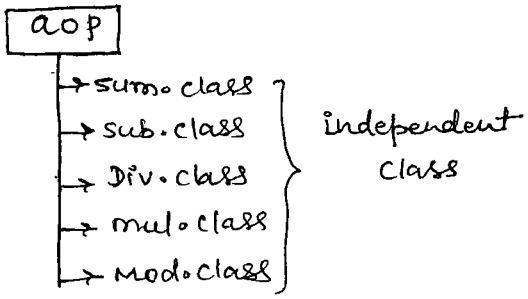
```
SOP ("w.r.t. I3 - AINC");
P3.I3 io3 = new P3.I3 ()
{
 Public void f1 ()
  {
   SOP ("f1 () - overridden- AINC");
  }
  Public void f2 ()
   {
    SOP ("f2 () - overridden- AINC");
   }
   Public void f3 ()
   {
    SOP ("f3 () - overridden- AINC");
   }
  };
        io3.f1 ();
        io3.f2 ();
        io3.f3 ();
   }
}
```

Model-III

Model-III makes us to understand there exist an <u>independent Package</u> and in which there exist collection of classes and interfaces and among them explicitly relationships are not applied (is-A, has-A, User-A)



Q. WAJP which will implements arithmatic operation ?



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

ACCESS MODIFIER

Access Modifier are applicable in two Level. They are –

(a) At class/Interface level.

(b) At Features of class level.

(a) Access modifier at class/Interface level –

Before the class/Interface we apply the Following access Modifier –

(1) Public [see Package - Phase-2 Point no.2]
(2) abstract [see abstract class]
(3) final
.....
(4) Public abstract
(5) Public final

Note:-
Combination of abstract final is not possible.

(b) Access modifiers at feature of class level –

Before the features of a class we write four types of access modifiers. They are –

1. private
2. Default [Not a keyword]
3. protected
4. Public

- If we are not using private, Public, Protected keyword then by default JVM will ~~treat~~ access modifiers as default.
- The purpose of access Modifiers is that where to access the features ~~and~~ where not to access the features

(OR)

- Access modifiers Provides features Controlling accessing Mechanism.

(OR)

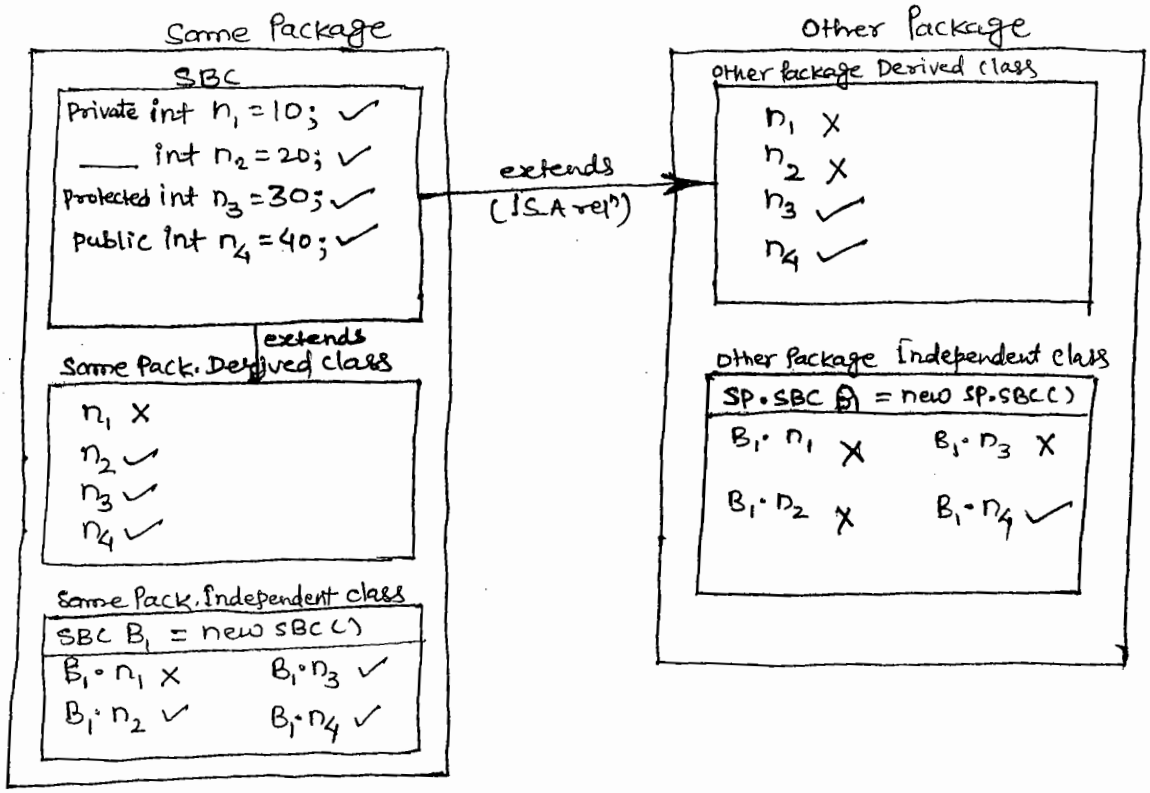
- Access Modifiers Makes us to understand how to access the Features within the Package and accross the Package betn Class-class, Interfaces-Interface, & Interfaces-class.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Rules for using access Modifier :-

The following diagram gives rules for using access Modifiers



Access Modifier Protection Matrix

Type of class / Type of Access Mod.	Same Pack. Base class	Same Pack. derived class	Same Pack. independent class	Other Pack. Derived class	Other Pack. Independent class
private	✓	X	X	X	X
default	✓	✓	✓	X	X
Protected	✓	✓	✓	✓	X
Public	✓	✓	✓	✓	✓

✓ ⇒ Represents Accessible/visible

X ⇒ Represents not accessible / Not visible

Syntax for declaring / Initializing variable along with Access Modifier

[Access Modifier] [Static] [final] datatyle V_1 [$=val_1$]. V_2 [$=val_2$] .... V_n [$=val_n$];

ex:-

```
public static final String sub = "Java";
  private int PIN = 4042;
 protected final int cardno = "xxxx";
   Static final String Inst = "sathya";
```

Syntax for defining a Method along with Access Modifier:-

[Access Modifier] [Static] [final] ReturnType MethodName (List of formal Parameter)

```
{
    Block of Stat(s) - B.Logic
}
```

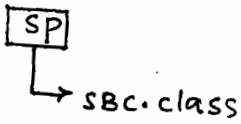
ex:-

```
public static void main (__)
{
  ___
  ___
  ___
}
```

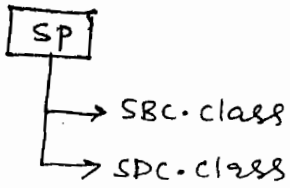
Q. WAJP which illustrate the rules of access modifiers.

```
// SBC.Java
Package sp;
Public class SBC
{ Private int n1 = 10;
          int n2 = 20; // Access Modifier is default
Protected int n3 = 30;
Public int n4 = 40;
Public SBC ()
{
   sop ("val of n1 =" + n1);
   sop ("val of n2 =" + n2);
   sop ("val of n3 =" + n3);
```

```
    sop ("val of n4 =" + n4);
  }
} //SBC
```




```
// SDC.Java
  Package sp;
  Public class extends SBC // is_A relationship within package
  {
    Public SDC ()
    {
    // sop ("val of n1 =" + n1); not accessible
      sop ("val of n2 =" + n2);
      sop ("val of n3 =" + n3);
      sop ("val of n4 =" + n4);
    }
  } // SDC
```



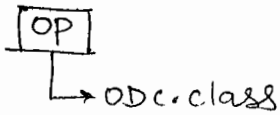

```
// SIND.Java
  Package sp;
  Public class SIND
  {
    SBC s1 = new SBC (); // has_A relationship ~~same class~~
    Public SIND ()                within ~~same~~ pack
    {
    // sop ("val of n1 =" + s1.n1); not accessible
      sop ("val of n2 =" + s1.n2);
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. write a Java main program which makes use of SBC, SDC, SIND of sp Package.

```
// SPD.Java
class SPD
{
 PSVM (string [] args)
 {
  SOP ("w.r.t. SBC");
  Sp.SBC S1 = new SP.SBC (); // 10 20 30 40
  SOP ("w.r.t. SDC");
   Sp.SDC S2 = new SP.SDC (); // 10 20 30 40 20 30 40
   SOP ("w.r.t. SIND");
    SP.SIND S3 = new SP.SIND(); // 10 20 30 40 20 30 40
```

```
// ODC.Java
 Package op;
 public class ODC extends sp.SBC // is_A relationship across the
                                    Package.
 {
   Public ODC ()
  {
   /* SOP ("val of n1 =" +n1); not accessible
      SOP ("val of n2 =" +n2); */
      SOP ("val of n3 =" +n3);
      SOP ("val of n4 =" +n4);
  }
}
```



```
//OIND.Java
Package op;
Package class OIND
{
 SP.SBC s1 = new SP.SBC();  // has-A relationship across Pack.
 public OIND()
 {
  /* sop ("val of n1 ="+s1.n1);
     sop ("val of n2 ="+s1.n2);
     sop ("val of n3 ="+s1.n3);   */
     sop ("val of n4 ="+s1.n4);
  }
 }
```

Q. WAJP which makes use of ODC & OIND of OP Package.

```
//OPD.Java
import op.ODC;
import op.OIND;
class OPD
{
 PSVM (String [] args)
 {
  sop (" w.r.t. ODC");
  ODC o1 = new ODC();  // 10 20 30 40   30 40
  sop ("w.r.t. OIND);
  OIND o2 = new OIND();  // 10 20 30 40   40
 }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Package The end :)

continue....

Exception Handling

- The aim of exception Handling is to build robust application.
- To develop any project as a part of real time, we need to choose a Language/Technology. By using this Language/Tech. we write some programs and we may get some types of errors and they are classified into three types.

They are :-

1. Compile time error.
2. Logical error.
3. Runtime error.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

1. **Compile time error :-**
   - These errors occurs during compile time due to syntaxes are not followed by programmers.
   - These errors solved by programmer at development level.

2. **Logical error :-**
   - These error occurs during execution time due to mis-interpretation/wrong representation of logic.
   - logical error always gives inconsistent/wrong result and these errors solved by programmers at development level.

3. **Run-time error :-**

   These error occurs during execution time when the application user enters "invalid input".
   - Runtimes errors always gives system errors messages which are understandable by only applications developer, But not Application users.
   - These errors solved by the programmers at implementation level. (which level there Appl^n user being use appl^n)

- The language like C/CPP/Pascal/COBOL etc and whose appl^n are treated as weak because these languages does not contain a facility called exception Handling.

- The language like Java and .Net and whose application are treated as robust (strong) because ~~they~~ these Language contains a Facility called exception Handling.

- To any project / programs, if any applications users enters any invalid input then by default we get system errors message. which are not recomonended to generating in s/w industry and highly recommended to generate user friendly error message by making use of exception handling facility.

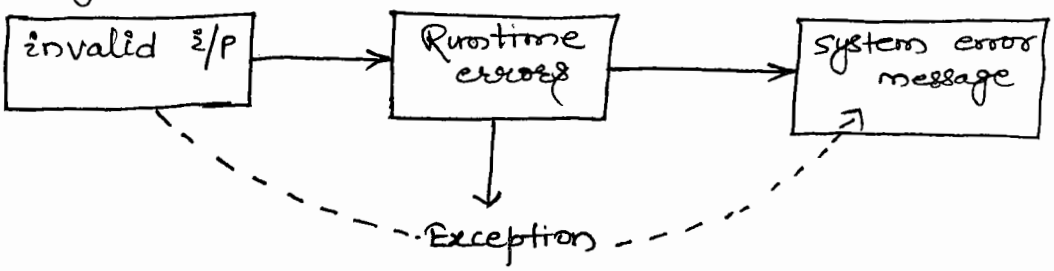
- Points to be remember in exception Handling:-

1. - when the Appl^n users enters invalid input then we get "Run time errors".

2. - Runtime errors always generate system error message.

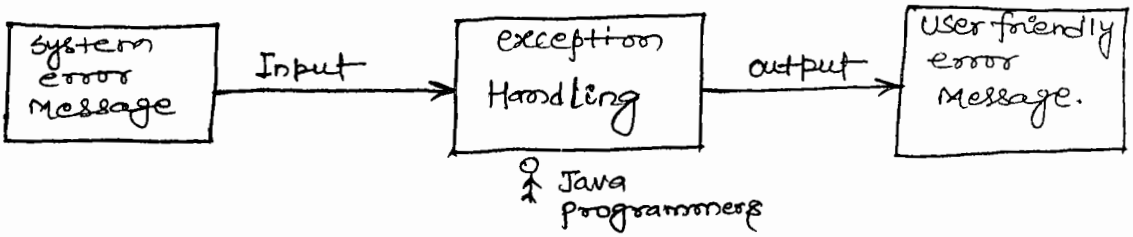
3. - Runtime errors of Java programs are known as "Exceptions".

4. - Every exception generate system error message



5. - whenever an exception occurs in the Java programs then programs execution is abnormally terminated, CPU control comes out of the program flow and JVM generates "system error message".

6. Def^n of execution Handling:-
The process of converting system error messages into user friendly error message is known as exception Handling



7 - Each & every operation / Task / Action must be performed with respect to object in Exception Handling.

S.E.M ⇒ System Error Message

- we know that whenever an exception occurs in the Java programs, program execution is abnormally terminated, cpu control comes out of the programs flow and JVM generates System Error Message.

- Generating S.E.M. is considered as an operation and it will be performed with an object in exception handling. Every object is requires a class and it is called "Exception class".

☆☆
- The purpose of exception class is to abnormally terminate the program execution when application users enters invalid input. By default an objects of exception class created by JVM and generates system error message and some exception class objects created by Java programmers and generates user friendly error message. which is one of the recommended process in software development.

- In general every problem occuring at run-time requires an exception class.

Types of exception :—

- we have two types of exceptions —
  (a) Predefined / Built-in exceptions
  (b) User / programmer / custom defined exception.

- Predefined exception developed by SUN MICROSYSTEM developer available as a part of Java s/w and whose role is deal with universal problems.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Some universal problems are :-

a.)- Division by zero problems (ArithmaticException)

b.) - Invalid number format (NumberFormatException)

c.)- Invalid bounds of the array
(Array Index out of Bounds Exception)

etc.....

- user defined exceptions developed by Java programmers available as a part of Java projects and whose role is deal with "common problems"

Some common Problems are :-

1. - Trying to enter invalid salary to an employee.
2. - Attempting to enter invalid PIN numbers in ATM based application.
3. - Attempting to withdraw access amount than existing.
4. - Attempting to enter invalid username and password

etc....

Types of predefined exception :-

predefined exceptions are classified into two types they are :-

(a) Asynchronous exceptions.
(b) Synchronous exceptions.

* Asynchronous exception :-

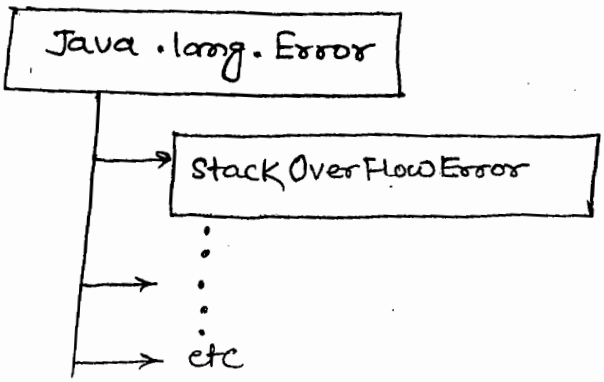
These exceptions deals with Hardware Problems and external problems.

ex:-

mouse / KBD / MotherBoard failure } H/W
memory Problem } Problems

Power failure } external Problems.

- At present asynchronous exceptions related API was not completely developed by SUN MICROSYSTEM developers (Partially developed) and there is no direct usage of these exceptions in real time, Due to partial development. for the future enhancement for dealing with all asynchronous exceptions, we have a pre-define super class called "Java.lang.Error".



*If the class ends with Error then those exception will be Asynchronous Exception.

Here, StackOverFlowError deals with Memory problem.

Synchronous Exception :-

synchronous exception are those which deals with programmatic runtime error. Synchronous exceptions are classified into two types they are :-

(a) checked exception.

(b) unchecked exception.

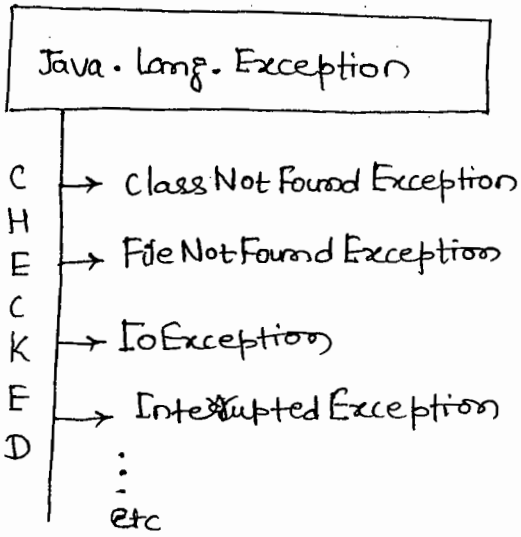
(a) Checked Exception :-

def^n 1 :- Checked Exception are those which are the direct sub class of Java.lang.Exception

or

def^n 2 :- The exception which are occuring at runtime will be showing them as error at compile time.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

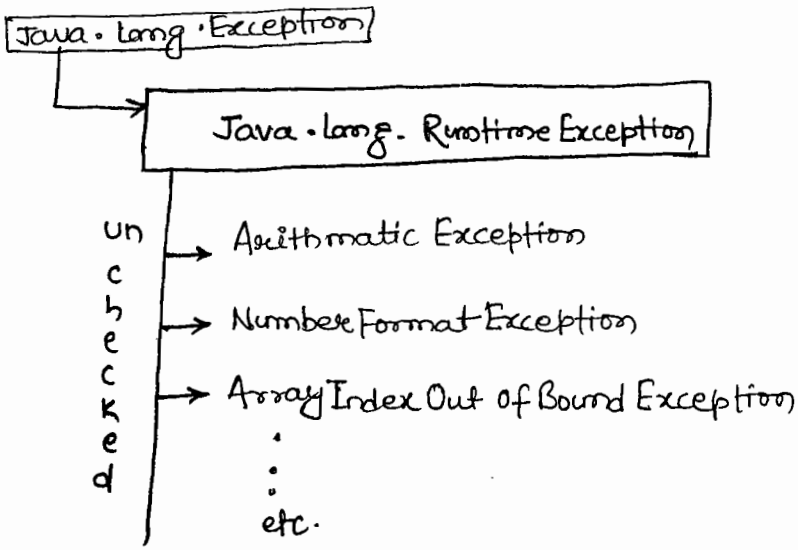


(b) Un-checked Exception :-

Def^n 1 :- Unchecked Exception are those which are the direct sub class of Java . lang . RuntimeException .

OR

Def^n 2 :- The exception which are occuring at runtime will be showing the runtime only but not at compile time .



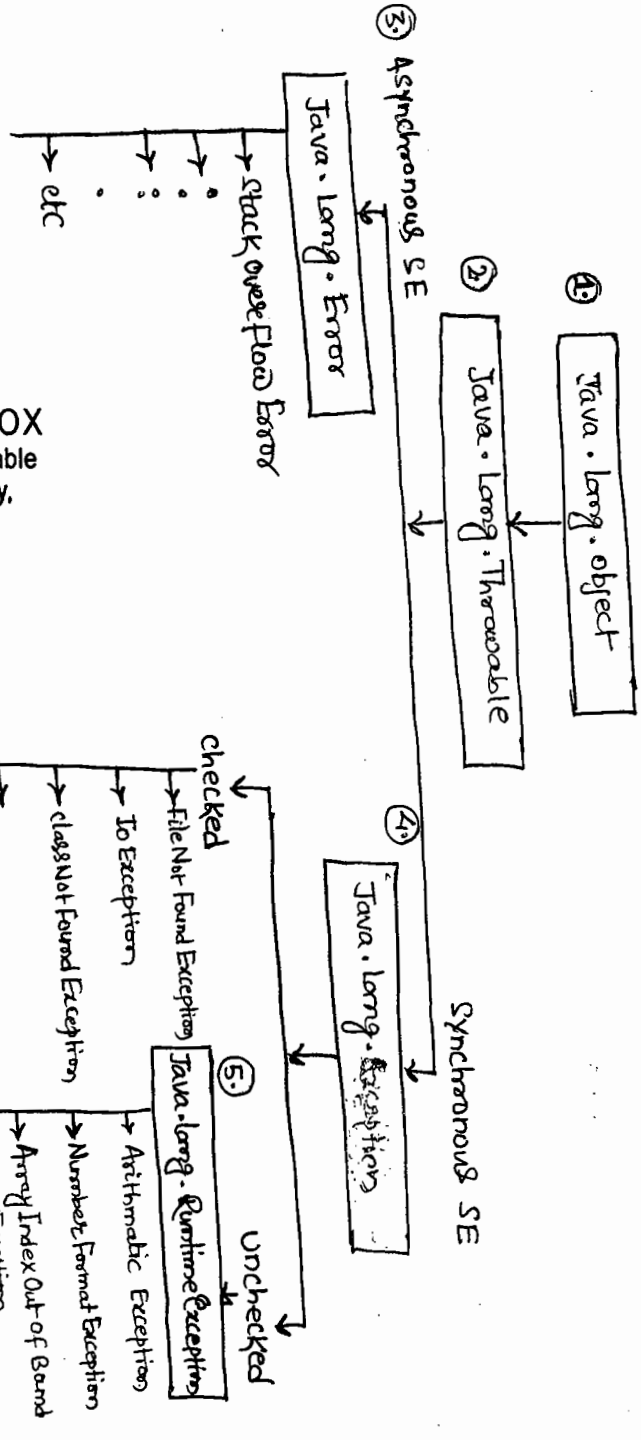
- Even Those Java . lang . Runtime Exception is subclass of Java . lang . Exception .

Runtime exception should not be consider as checked exception .

Exception Handling Hierarchy chart :-

Hierarchy chart makes us to understand how to features are inherited from top most Base class to intermediate Base class and from intermediate Base class to Bottom most ~~Base~~ Derive class.

The following diagram makes us to understand the list of classes available in exception handling.



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

1.) - Java.Lang.object is one of the implicit predefine super class for all the classes in Java and its provides Garbage collector facility for unused memory space.

2.) - Java.Lang.Throwable is of the super class for all the exceptions in Java. The purpose of Java.Lang.Throwable is that to decide what type of exception occurs in the Java Program (either Asynchronous or Synchronous).

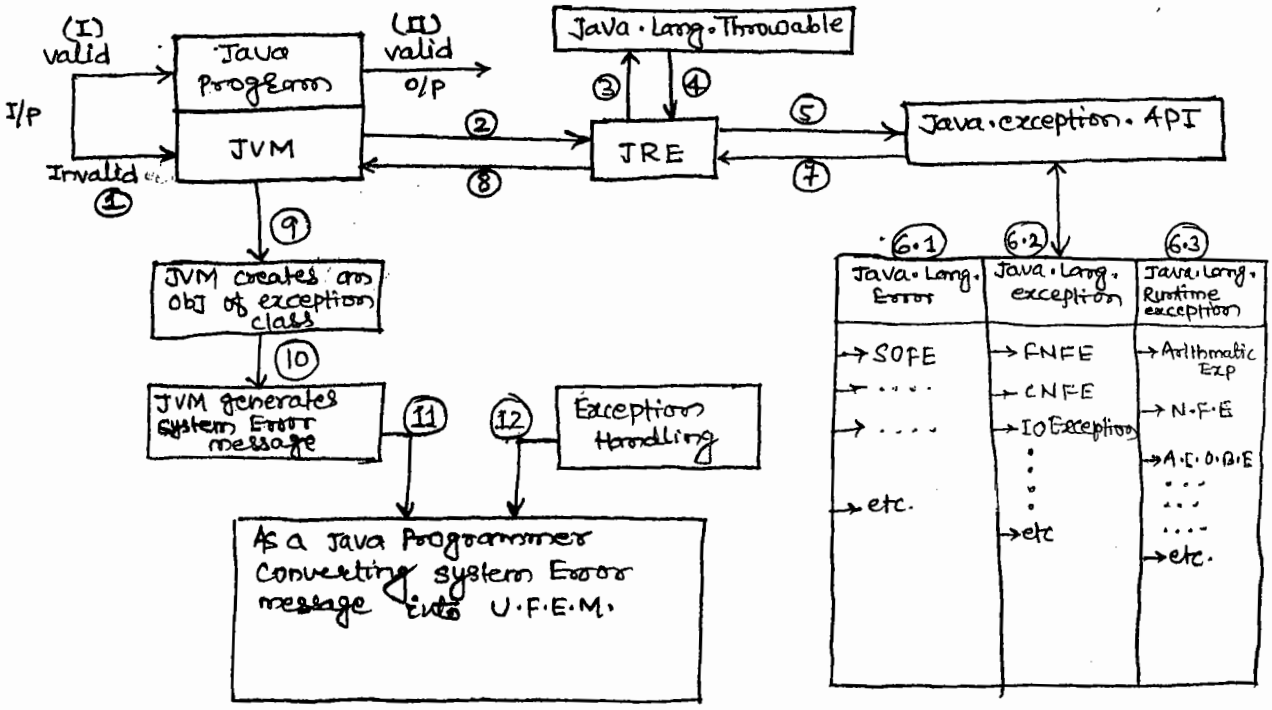
3.) - Java.Lang.Error is the super class for all Asynchronous exception.

4.) - Java.Lang.Exception is the super class for all Synchronous checked exceptions.

5.) - Java.Lang.RuntimeException is the super class for all synchronous unchecked exceptions.

- Even through exist multiple exception in Java program; JVM can process only one at a time: By created an object of exception subclasses. ~~Hence whenever Exception occurs in Java~~

Internal Flow of Exception Occurance:-



The following diagram gives internal flow of occurance.

Steps :-

I. Java application user enters valid input to the Java Program.

II. JVM can process Java prog. with valid input & gives valid output.

<1.> Application user enters invalid input to the Java Program.

<2.> JVM can't process Java program with invalid input and it contact to JRE for getting appropiriate exception sub-class.

<3> JRE communicates with Java.Lang.throwable for finding what type of exception occures in Java Program.

<4.> Java.lang.throwable is one of the predefine superclass for all the exception in Java and it can find what type of exception occurs in Java Program, and the same message gives JRE.
(JRE = JVM + Java API)

<5.> JRE communicates with Java exception API by handovering types of exception occures in Java Program, for obtaining exception sub-class.

<6.1, 6.2, 6.3> :-
Java exception API executes either 6.1 (Java.lang.Error For ASE) or 6.2 (Java.lang.Exception for Synchronous checked) or 6.3 (Java.lang.Runtime exception for synchronous unchecked) and obtains exceptions of class.

<7> Java exception API gives appropiriate exception of class to JRE.

<8> JRE gives exception subclass to JVM which was received from Java exception API.

<9.*> JVM creates an object of exception sub-class
Ex:- If we enter 10 & 0 as an input Div.Java Program JVM will create an object of Arithmatic exception.

Arithmatic Exception ae = new Arithmatic Exception ("/ by zero");
(ae: obj; new: Runtime memory allocator; "/ by zero": nature of message)

<10.> After creating an obj. of exception subclass JVM will generate system error message.

Tech. def. of Exception – An Exception is an obj. runs at runtime which describe the nature of message. The nature of mssg can be either SEM (or) UFEM

Ex:- The system Error message for the exception which is occures in exp. (9) step; is shown below —

SEM (System Error Message)

```
Example in thread main . . . . . . . . . .
Java.lang. Arithmatic Exception : 1 by zero . . . . . .
```

+ Industry is not recommended to generate SEM & highly recommended to generate U.F.E.M (User Friendly Error Message)

<11.12> As a Java Programmer by using exception handling we can convert system Error messages into user friendly Error messages.

Ex:- The User friendly Error message for the system Error message which is occured into example of step (10) is shown below:—

UFEM :→ "Don't Enter zero for Denominator"

Handling the Exception:—

Handling the Exception is nothing but it converting SEM into User friendly Error messages.

To do process, as a part of exception handling we have five keywords they are:—

1. try
2. catch
3. Finally
4. throws
5. throw

· Syntax for handling the exceptions :-

```
try
{
 Block of stat (s) causes problem at Runtime
}
catch (type of exception obj1)
{
 Block of stat(s) - provides UFEM
}
catch (type of exception obj2)
{
 Block of stat(s) - Provides UFEM
}
```

```
:
:
catch ( Type of exception obj n)
{
 Block of stat (s) - UFEM
}
finally
{
 Block of stat (s) which will execute compulsory
}
```

1) **try :-**

(a.) It is the block in which we write the block of statements causing problem at run-time or what are all the block of stat. generating exception, those stat. must be written within try & catch block. This block is also known as "exception monitoring block".

(b.) If any exception occurs in try block, than JVM control comes out of try block and appropriate catch block will be executed.

(c.) After executing appropriate catch block JVM control never goes to try block to execute rest of statement even if we use return statement in catch block.

```
try
{
}catch
{
 return();
```

(d.) Each & every try block must be immediatly follow by catch block.

Programmatically intermediate stat. are not permitted b/w try & catch block.

```
try
{
}
catch()
compulsary
```

(e.) E & E try block must contain atleast one catch block. Programmatically it is highly recommended to write multiple catch blocks for generating multiple Userfriendly error messages. to make the application robust.

(f.) In some of the circumstances one try block can contain another try block i.e. inner/nested try block can be permitted.

```
try
{
 try
 {
  }catch()
 }
 catch()
}
```

nested/ Inner try block

(ii) <u>Catch</u> :-

(a) It is the block in which we write the block of statements which will provides U.F.E.M. In other words catch block will seperates SEM & generates UFEM. This block is known as <u>exception Handler block</u>.

(b) Catch block will be executed by JVM provided an exception occurs in try block.

(c) At any point of time JVM will execute only one catch block even their exist multiple catch block.

(d) In the catch block as a Java programmer we declare an object of exception subclass and implicitally which is referenced by JVM.

ex:-

```
try
{
  ==
  int a = 10/0 ;
  ==
}
catch ( Arithmatic Exception ae )
{
  ==
}
```

Decl. done by J.P (Java Programmer)

= new Arithmatic Exception ("/ by zero");

ref. by JVM done

(e) A catch block must be followed by finally block (if at all we write finally block programmatically we shouldn't write intermediate stat betn catch & finally block.).

(f) In one catch block one can also write try & catch block.

<iii> finally*:

(a) It is the block in which we write the block of statements which will relinquise (release/close) the resources (files/DB) which are obtained in try block.

(b) finally block will execute compulsory.

(c) writing finally block is optional.

(d) It is highly recommended to Java programmer to write one finally block per Java program.

(e) In some of the circumstances a finally block can contain try & catch block.

(f) If at all we write finally block, it should be followed after catch block (if catch block present) or it can be followed after try block (if catch block not present).

Q. write a Java Program which illustrate the concept of try & catch, finally concept.

(write a Java Prog. which will cal. div of two no's & handle the appropiriate exception)

```
//Div2.Java
class Div2
{
 public static void main (String[] args)
 {
  try
  {
   String s1 = args[0];
   String s2 = args[1];
   int x = integer.parseInt(s1);
   int y = integer.parseInt(s2);
   int z = x/y;
   System.out.println("Div in Div2 =" +z);
  }
  catch (Arithmatic Exception ae)
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
{
 System.out.err.println ("don't enter zero for den...");
}
catch (NumberFormatException nfe)
{
 System.err.println ("don't enter alpha-numeric value");
}
catch (ArrayIndexOutOfBoundsException ab).
{
 System.err.println ("enter two value");
}
finally
{ System.out.println ("I'm in finally block");
}
} }
```

Case 1:-

If (an exception occurs) then
⇒ A part of try Block will execute.
⇒ An appropriate catch Block will execute.
⇒ A finally Block will execute (if we write).

Case 2 :-

If (an exception doesnot occurs) then
⇒ A complete try Block will executes.
⇒ A finally will executed (if we write)

<iv.> throws :-

In the Context of exception handling we have two types of methods. They are -

1. specific method
2. Common method

(1.) specific method -

specific method always deals with specific req. of individual Programmers.

If any exception occurs in this method then specific method developer must handle such exception by writting try & catch block. and one should not use throws Keyword because nobody call specific method.

② Common method —

A common method always deals with common requirements which are suitable for all the programmers.

If any exception occurs in common method body then such exceptions should not be handled and those exception must be send to specific method by making use of throws keyword

→ The purpose of throws keyword is that it describe/express the exceptions which are occuring in common method body

or

- Throws keywords are always gives an indication of specific method to place the common exception method within try & catch block for generating user friendly error message instead of generating system error messages

- The place of using throws keyword is always as a part of common method heading.

- It is always recommended to Java Programmer before using the predefine methods (common method). Check the prototype.

- If they contain throws keyword then handle the exception otherwise need not to handle.

Rules of throws keyword —

1. when throws keyword is throwing checked exception then it is mandatory to Java Programmer to handle checked exception of placing the common exception methods within the try & catch block otherwise we get compile time error.

2. when throws keyword is throwing unchecked exception then it is optional to Java programmer to handle unchecked exception of placing the common exception and not need to write try & catch block

3. Irrespective of type of exceptions thrown by throws keyword, it is always recomended to Java Programmer to handle such exceptions by writing in try & catch block for eleminating compile time error message & provide UFEM.

Syntax of throws :—

Ret Type method Name (List of formal Parameter if any) throws <cls name$_1$>
. . . . . .
<cls name$_n$>

{

Block of statement – Business Logic

}

Here <cls name$_1$>, ... <cls name$_n$>, name of exception sub-classes & they may be checked or unchecked.

Q. Write a Java Program which listed the concept of throws keyword which is used as a part of predefine common exception method.

Let us consider the following –

Java.lang.Integer

Public static int parseInt (String) throws numberFormatException

Here - parseInt method is throwing predefine unchecked exception called NumberformatException. It is recommended to place within try & catch block for generating user friendly Error message.

```
// SathyaProg1.java
import java.util.Scanner;
class SathyaProg1
{
Public static void main (String [] args)
{
 Scanner s = new Scanner (System.in);
 SOP ("Enter your marks:");
 String marks = s.nextline();
 try
 {
   int im = Integer.parseInt (marks);
   system.out.println ("marks=" +im);
 }
```

```
    Catch (NumberFormatException nfe)
    {
     system.err.println ("don't enter alphanumerical for marks");
    }
    Finally
    {
     sop ("I'm from finally block");
    }
  }
}
```

Q. write a Java program which illustrate the concept of throws keyword which is used as part of user define common method.

```
// SathyaDiv.java
Package SP;
Public class SathyaDiv
{
 public void division (String s1, string s2) throws
 ArithmaticException, NumberFormatException
 {
   int a = integer.ParseInt (s1);
   int b = integer.ParseInt (s2);
   int c = a/b;
   sop (" div in SathyaDiv =" +c);
 }
}
```

Compile the above program

```
Javac -d . SathyaDiv.java
```

space

In SP.SathyaDiv class the division method; is one of the programmer define common exception method and it must be always written within try & ~~catch~~ catch blocks.

Q: write a java main prog. which makes use of division(--) method of sp.SathyaDiv class

```
// Sathyaprog2.java
import java.util.Scanner;
import sp.SathyaDiv;
class SathyaProg2
{
  PSVM (String[] args)
{
  Scanner s = new Scanner (System.in);
  SOP ("Enter first value:");
  String x1 = s.nextline();
  SOP ("Enter Second value");
  String x2 = s.nextline();
  SathyaDiv sd = new SathyaDiv();
  try
{
sd.division(x1, x2); // Programmer-def common exception method
}
 catch (ArithmaticException ae)
{
  SOP ("don't enter zero for den...");
}
 catch (NumberFormatException nfe)
{
   system.err.println ("don't enter alphanumeric value:");
}
 finally
{
 SOP ("I'm from finally block:");
}
}
}
```

* Re-throwing the Exception

Def:- Method 1 is throwing an exception to method 2. method 2 is again throwing method 1 related exception to method 3. With respect to method 2; method 1 related exception is called Re-throwing Exception.

Ex:- In Java API, sp.sathyaDiv and sathyaProgrammer2 classes the parseInt method () is throwing NumberFormatException to Division (–, –) method. The Division (–, –) method is again throwing NumberFormatException of ParseInt ( ) method is throwing main (–) methode (method 3); w.r.t to division (–, –) method NumberFormat Exception of parseInt ( ) method is called re-throwing Exception.

Note :-

- out & err are the objects of printstream class and they are created as a static datamember in system class.
- out object is always used for displaying soft outputs of Java program and err object is always recommended to use to display abnormal terminated messages for making use of error handling services.
- In is an object of input stream created as static data member in system class. in objects provides an environment to read the data from keyboard.

(V.) Throw :—

The purpose of throw keyword is that it will carry the created exception from method body & handovering to throws keyword.

OR

Throw keyword is used for hitting/generating the exceptions which is occuring as a part of common method body.

- The place of using throw keyword is always as a part of method body.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Rules for using throw keyword —

1.) when throw keyword is hitting checked exception then it is mandatory to the programmer to write throws keyword otherwise we get compile time error.

2.) when throw keyword is hitting unchecked exception then it is optional to programmer to write throws keyword.

3.) Irrespective of the type of exception throw keyword is hitting. Then it is highly recommended to write throws keyword. (Just for making the specific method developers to understand what type of exception are thrown by the common methods)

Syntax 1 :—

```
<class name> <obj> = new <clsname (message)> ;
        throw obj ;
```

OR

```
throw (new <clsname (message)>);
```

example -

```
ArithmaticException ae = new ArithmaticException ("don't enter zero
                                                   for den...");
throw ae ;
```

Syntax 2 :-

```
ret-type methodName (list of formal Params if any) throws
<clsname-1>, <clsname-2>.... <clsname-n>
{
  if (test cond1)
  {
   <clsname-1> <obj1> = new <clsname1 (message)>;
      throw obj1 ;
  }
  if (test cond2)
  {
  <clsname2> <obj2> = new <clsname2 (message)>;
  }
  ========
}
```

<cls name 1>, <cls name 2> .... <cls name n> represents list of exception for class and they may belongs to either checked or unchecked.

Q: write a java program which listed the concept to throw keyword. (How do you recreate an obj of arithmatic Excep with our own user friendly Error message)

```
// SathyaDiv1.java
Package sp1;
public class SathyaDiv1
{
 public void division (String s1, String s2) throws
 ArithmaticException, NumberFormatException
{
   int a = Integer.parseInt (s1);
   int b = Integer.parseInt (s2);
  if (b==0)
  {
  ArithmaticException ae = new ArithmaticException ("don't enter zero
                                                     for deno. . . .");
  throw ae;
  }
 else
 {
   int c = a/b;
   System.out.println ("div in SathyaDiv=" + c);
 } // else
} // division (-,-)
} // sathyaDiv1
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

→ compile the above programs —

```
javac -d . SathyaDiv1.java
```

SP1 → sathyaDiv1.class

In SP_1 SathyaDiv1. class the Division (-,-) is one of the common exception method and it must be placed with in try and catch block.

Q. Write a java main program which will display recreated message of arithmatic Exception by making use of ~~SP.Sathya Div1~~ SP.SathyaDiv1.

```
//SathyaProg3.Java
Import Java.util.Scanner;
Import SP1.SathyaDiv1;
class Sathyaprog3
{
 Public Static void main (String [] args)
 {
  Scanner s = new Scanner (System.in);
  SOP ("enter first value");
  String x1 = s.nextLine();
  SOP ("enter Second value:");
  String x2 = s.nextLine();
  SathyaDiv1 sd = new SathyaDiv1 ();
  try
  {
   sd.division (x1, x2); // Programmer-def common excep. method
  }
   catch (ArithmaticException ae)
   {
   System.err.println (ae); // Java.lang.A...E: don't enter zero for deno....
   }
   catch (NumberFormatException nfe)
   {
    System.err.println (nfe)
   }
   finally
     SOP ("I'm from finally block");
   }
 }
}
```

- write

```
//NameValidation.java
Package np;
public class NameValidation.
{
  public void validationName (String name) throws
  NumberFormatException
  {
    boolean found = false;
   for ( int i = 0; i<name.length (); i++)
   {
    if ( character.isDigit (name.charAt (i)))
   {
     found = true ;
     break;
   }
  } // for
  if (found)
   {
    NumberFormatException nfe = new
    NumberFormatException ("Name should not contain digits");
    throw nfe;
   }
   else
    {
     System.out.Println ("your name is valid");
    }
   }
  }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

compile the above program

```
javac -d . NameValidation.java
```

np
↓
NameValidation.class

Q. write a Java Program make use of np.NameValidation for validating the names.

```
// SathyaProg4.java
import java.util.Scanner;
import np.NameValidation;
class SathyaProg4
{
 psvm (String [] args)
{
  Scanner S = new Scanner (System.in);
  S.op ("enter your name:");
  String name = S.nextLine ();
  NameValidation no = new nameValidation ();
 try
  {
   no.validateName (name); // Programmer def, common exception method
  }
  catch (NumberFormatException nfe)
  {
   system.err.println (nfe);
  }
  finally
  {
   S.O.P ("I am from finally block);
  }
 }
}
```

Differences between throw & throws

throw	throws
- It is a keyword used for hitting/generating the exception which are occuring as a part of method body.	- Throws is a keyword which gives an indication to the specific method to the place the common exception method under try & catch block for generating user friendly error message.
- The place of using throw keyword as a part of method body.	- The place of using throws kwd as a part of common method heading.
- Irrespective of type of exception hit by throw keyword, it is highly recommended to write throws keyword as a part of method heading.	- when we use throws keyword as a part of method heading it is optional to use throw keyword as a part of method body (Applicable for predefine exception & rethrowing the exception).

V.V.I

→ * **No. of approaches for finding detail about un-known exception:-**

when we write a Java prog. there is a possibility of many no. of exceptions and these exception may not be knowing to Java Program, and it results in system error messages.

- In order to find details about un-known exception we have three approaches:-

(a.) By using an object of Java.lang.Exception / Throwable

(b.) By using PrintStackTrace ()

(c.) By using getMessage ()

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

(a.) By using an object of Java.lang.Exception/Throwable –

According to concept of dynamic Binding an object of Java.lang.Exception/Throwable gives the following detail –

1. Name of the unknown Exception
2. Nature of the message.

ex –

```
try
{
  int a = 10/0;
}
catch ( Throwable t)
{
  sep (t); // java.lang.A....E: / by zero
}
```

= new Arithmatic Exception("/ by zero");

ref. by JVM

Name of exception

Nature of message

(b.) By using PrintStackTrace ()

It is the one of the predefine() it is present in Java.lang.Throwable & inherited into both Java.lang.error & Java.lang.exception.

– By using PrintStackTrace () we can find the following detail –

1. Name of the unknown Exception
2. Nature of message.
3. Line no. where the exception occurs

The Prototype of PrintStackTrace() is given below

public void PrintStackTrace ()

Since this method is returning void type ; It should not be accessed by sop & sep and it should be called directly w.r.t. corresponding class object.

X.java

```
try
{
1: ___
2: ___
3: int a = 10/0;
4: ___
} catch ( Throwable / Exception t ) = new ArithmaticException("/ by zero");
                                      ref. by JVM
{
  sep (t.printStackTrace()); // Invalid
  t.printStackTrace(); // java.lang.A...E: /by zero at X.java : 3
```

name of exception — nature of message — Line no. where exception occure

(c.) By using getMessage();

It is one of the predefine method present in java.lang.Throwable and inherited into both java.lang.err and java.lang.exception with is method we can find only —

(a) nature of message

The prototype of this method is given as —

```
Public String getmessage() ;
```

ex —

```
try
{
  int a = 10/0;
} catch ( Throwable / Exception t ) = new ArithmaticException("/ by zero)
                                      ref. by JVM
{
  sep(t.getmessage()); // /by zero
}
```

nature of message

V.V.9

*Development of User/custom/secondary/Programmer defined exception

Def:- User defined exception are developed by Java programmers, available as a part of Java project and whose role is to deal with common problem.

Some of the common problems are —

1. - Attempting to enter negative salery, ages, etc.
2. - Attempting to deposite negative amt. in the balance of account
3. - Attempting to withdraw excess amt. than existing.
4. - Trying to enter invalid username/password.
5. - Trying to enter invalid PIN in ATM based Appl^n.
6. - Trying to enter invalid no. of characters for passport form in sign-up form.

need -

The purpose of defining our own exception class is that to abnormaly terminate the program execution, when appl^n users enter invalid input.

- what are all the guidlines are followed by sun microsystem dev. for development of pre-defined exception;
- same guidelines followed by Java programmer, for development of user-defined exception classes —

Steps/Guidelines for development of user-define exception class:-

1. - choose the Package Name for Placing user-define exception-sub-class for common access and ensure that Package statement must be 1st executable statement.
2. - choose an appropriate user define class and ensure that whose modifier must be public.
3. - whenever the class is selected in step 2 must extends either Java.Lang.exception OR Java.Lang.RuntimeExceptiion for inherating the program Execution abnormally when the application user Enter invalid input. Hence step 2 class is called user define Exception Sub-class.
4. Each & Every user define Exception sub-class must contain single parametrized constructor by taking string as a parameter which will describe the nature of the message.
5. Each & every user define Exception sub-class parametrized constructor must call the parameterized constructor of either Java.Lang.Exception (OR) Java.Lang.RuntimeExceptiion by making use of super(msg); Here (msg) represents string parameter which describe Nature of Message.

(The reason of writing super(msg) is if specific Exception occurs is the Java Program and it is not handled by Program then Specification Exception Related Parametrized Exception must call parametrized Exception of either Java.lang.Exception (OR) Java.lang.RuntimeException.)

⑥ - whatever the user defined Exception subclass we are placing in the package it must be given as a file name with an Extension .Java.

- Develop user define Exception subclass for dealing with negative salary of an employee.

```
// NsalException.Java
Package Sathya.lang;
Public class NsalException extends Exception
{
  Public NsalException (String s)
  {
    Super (s);
  }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

compile the above program —

```
Javac -d . NsalException.Java
```

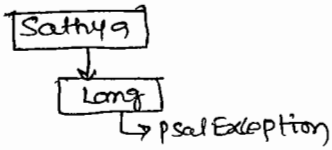
Sathya → lang → NsalException

Q. Develop user define Exception subclass for dealing with Positive salary of an employee.

```
// PsalException.Java // 6
Package Sathya.lang; // 1
public class PsalException extends RuntimeException // 2 // 3
{
  Public PsalException (String s) // 4
  {
    Super (s); // 5
  }
} // PsalException - dev of user def exception Sub class
```

for compiling the Program -

javac -d . PsalException.java



✱ No. of Phases required for development of user/programmer defined exception based application:-

To develop user-defined exception based appl^n in the industry we follow three phases they are —

(1.) Development of user-defined Exception sub-class.

(2.) Development of user-defined common class.

(3.) Development of user-defined specific class

Phase-1: Development of user-defined Exception sub-classes.

To develop phase-I related class we must follow 6 guideline (see previous pages).

ex- NsalException.java, PsalException.java

(see previous pages)

Phase-2 - Development of user-defined Common class

Phase-II classes always deals with common requirements of most of the programmers.

This phase related classes makes use of phase-I related classes for hitting the exception. when appl^n users enters invalid input.

Q. write a java Program which will check salary of every employee & hit the appropriate exception if required.

```
// Emp.java
package org.igate;
import sathya.lang.NsalException;
import sathya.lang.PsalException;
public class Emp
{
public void decidesal (String s) throws
NsalException, PsalException, NumberFormatException
{
int sal = Integer.ParseInt (s);
if (sal <= 0)
```

```
NsalException no = new NsalException ("invalid sal");
throw (no);
}
else
{
PsalException po = new PsalException ("valid sal");
throw (po);
} // else
} // deside sal(-)
} // Emp - dev. of user-def. common class.
```

- Compile the above program
  javac -d . Emp.java
- In org.Igot.Emp class the decidesal(-) is one of the userdefined common exception method and it must be return always within try & catch block

Phase-3: Development of user-define Specific class :-

This phase classes always deals with specific requirement of individual and they always makes use of phase-2 classes and handles the exception if required.

Q: write a java prog. which will check the sal. of sathya emp & handle the appropriate exp. if required.

```
// SathyaEmp.Java
Import java.util.scanner;
import org.igate.Emp;
import Sathya.Lang.NsalException;
import Sathya.lang.PsalException;
Class SathyaEmp
{
P.S.V.M. (String [] args)
{
Scanner S = new Scanner(System.in);
SOP ("enter the sal. of sathya Emp");
String sal = S.nextLine();
Emp eo = new Emp();
```

```
try
{
  eo.decideSal (sal);
}
catch (NsalException no)
{
 sop ("don't enter -ve/zero sal for an emp);
}
 catch (PsalException no)
{
 s.e.p(" Good sal, try for best");
}
 catch (NumbeFormatException no)
{
  s.e.p ("don't enter alpha-numeric sal");
}
 catch (Exception no)
 {
  s.e.p.(no);
 }
 Finally
  {sop ("I'm from finally block");
  }
 }
}
```

Q. Develop user define exception based application for dealing with division by zero problem by developing our own exception subclass without using ArithmeticException class.

Phase-I

```
//SathyaArithmeticException.java
Package spa;
Public class sathyaArithmeticException extends Exception
{
 Public SathyaArithmeticException(String s)
 {
   super(s);
 }
} // sathyaArithmaticException - dev of U.D.E.S.C.
```

compile the above program -

```
javac -d . SathyaArithmeticException.java
```

Phase-II

```
//sathya.java
Package sat;
import spa.sathyaArithmeticException;
Public class Sathya
{
Public void division (int a, int b) throws SathyaArithemeticExcep.
{
        if (b==0)
        {
          throw (new SathyaArithmeticException("/ by zero"));
        }
    else
    {
      int c=a/b;
      SOP ("division in sathya"+c);
    }
  }
} // sathya - dev. of U.D.E.S.C.
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Phase-III

```
//AED.Java
import java.util.Scanner;
import sat.sathya;
import spa.SathyaArithmeticException;
class AED
{
  P.s.v.m.(String [] args)
{
  Scanner s = new Scanner(System.in);
  sop("Enter first value:");
  int x = Integer.ParseInt(s.nextline());
  sop("Enter second value:");
  int y = Integer.ParseInt(s.nextline());
     Sathya S1 = new Sathya();
  try
  {
     S1.division(x,y);
  }
     Catch(SathyaArithmeticException ae)
     {
      sep("don't enter zero for deno..");
      }
      catch(Exception e)
      {
       sep(e);
      }
      finally
       {
        sop("I'm from Finally block");
       }
      }
    }
```

Q. Develop a U.D.E. based appl^n for user authentication (user authentication nothing but only valid name and password)

```
//InvalidLoginException.java
  Package IP;
  public class InvalidLoginException extends Exception
  {
   public InvalidLoginException (String s)
   {
    super(s);
   }
  } // invalid loginException - dev of UDESC

// validLoginException.java
  Package VP;
  public class validLoginException extends RuntimeException
   {
    public validLoginException (String s)
    {
     super(s);
    }
   }
// Auth.java
  Package org.tcs;
  Import VP.ValidLoginException;
  Import IP.InvalidLoginException;
  public class Auth
  {
   public void login (String un, String pwd) throws
        validLoginException, InvalidLoginException.
   {
    if ((un.equalsIgnoreCase("sathya")) && (pwd.equals("java")))
    {
     throw (new validLoginException("Login success"));
    }
    else
    }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
else
 {
   throw(new InvalidLoginException("Login Denied"));
 }
}
}
```

Compile- `Javac -d . Auth.java`

Phase-III

```
// SathyaUser.java
import org.tcs.Auth;
import vp.validLoginException;
import ip.InvalidLoginException;
Import java.util.Scanner;
class SathyaUser
{
  P.S.V.M (String [] args)
 {
  Scanner s = new Scanner(System.in);
   SOP ("enter User Name");
   String un = s.nextline();
   sop ("enter Password");
    String pwd = s.nextline();
    Auth ao = new Auth();
   try
  {
    ao.login (un, pwd);
  }
  catch (validLoginException vo)
  {
   sep ("Ur user name/pwd is correct & proceed");
  }
  catch (InvalidLoginException vo)
  {
  sep ("Ur user name/pwd is incorrect & try again");
  }
  catch (Exception vo)
    {
    } sep (vo);
```

```
finally
  {
    sop ("I'm from finally block");
  }
}
}
```

- New features in Exception Handling as a Part of JDK 1.7/1.8

1) multi exception catch block :-

- The purpose of this features is handling multiple specific Exception in a single catch block.
- With this feature along with one user friendly error message we can also give other user friendly error messages in advance.

Syntax :-

```
try
{
 Block of stat(s), causes Problem at runtime
}
catch (exception1 | exception2 | .... | exception n)
{
 Block of stat(s), which provide user friendly
                    error messages
}
```

ex:-

```
try
{
 ----
 ----
 ----
} catch ( ArithmaticException | NumberFormatException )
  {
    sop ("Don't enter zero for deno...");
    sop ("Don't enter alphanumeric value");
  }
```

(2) try with resource management
(OR)
Automatic resource mangement (ARM) :-

The advantage of this features it relinguise the resources automatically in the context of try block only without explivly writing finaly block.

- ~~If any~~ programatically if any exception occurs in the try block then resource are relinguise which are obtained in try block and later it comes to appropiate catch block and it is not neccessary to write finally block.

syntax :-

```
try ( Block of stmt(s) . provide
      Logic for resource gathering )
  {
    Block of stmt(s) — causes
  }
    catch (—)
    catch (—)
      :
```

Example :-

```
try ( FileInputStream fis = new FileInputStream ("sathya data");
      FileOutputStream fos = new FileOutStream ("sathya data") )
  {
    ——
    ——
    ——
  }
  catch (—)
  catch (—)
```

- In any exception occurs in

Conclusion :- since, try block automatically relinguise in the resources & the exception occurs, it is called automatically resource management block.

Collection Framework (Java.util.*)

09 oct-15.

- The real-time implementation of collection framework is to develop high performance based application.

Definition:-

- A collection framework is one of the standard mechanism which allows us to group similar type of elements, different type of elements or both type of ele. in single variable with dynamic size; this single variable is called collection framework. variable.

- The default env. of Java prog. says the values are sending or receiving between main memory & secondary memory, between multiple programs in the same project & b/w multiple remote machine where, they are communicating through network. To overcome this time consuming process in Java programming we have an additional service called collection framework.
  The concepts sends all at once & receive all at once irrespective of the type of Application.

Q. what are the differences between arrays & collection framework

arrays	collection framework
(a) An array is a collective name given to a group of consecutive memory location, which are all records by similar type of values.	(1) A coll^n fwk is one of the standard mechanism which allows us to group similar type of ele, different type of ele. or both type of ele, in single variable with dynamic size, this single variable is called collection framework variable.
(b) Arrays concept allows us to organize only Homogenous val. but not Heterogenous val.	(2) collection framework allows us to organize both Homogeneous & Heterogeneous val.
(c) Arrays concept contains fixed size of element.	(3) collection framework contain dynamic size.
(d) Arrays are non-serializable.	(4) collection framework is serializable.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. when array of obj. of java.lang.object is allowing all types of value then what is need of collection framework? ~~even~~

- Even through an array of object of java.lang.object is organizing multiple values of same type or different type, we prefer to go for collection Framework because of following limitation:—

a.) Array of object of java.lang.object contains fixed size.

b.) Not belongs to serializable.

c.) Unable to perform dynamic insertion.

d.) Unable to display all the elements at once.

Advantages of Collection framework :-

1. Collection framework provides high performance.
2. collection framework provides Extendability. (depends on the incoming flow of data; the size of collection framework variable is increases)
3. collection framework provides Adoptability. (The content of one collection framework variable can be added to another collection framework variable either in beginening, ending or middle)
4. Collection framework is one of the Algorithmic oriented.
   - 4.1 :- CFW provides Built in sorting Technique.
   - 4.2 :- CFW provides Built-in searching Technique.
   - 4.3 - CFW provides preliminary services of Data structures such as stack, queue, linked list, trees . . . etc

- To Deal with CFW programming we must import a Package called java.util.*

Types of collection Framework :-

- In the initial days of Java Programming the Concept of collection framework was known as data structure.
- The data structure concepts are able to fullfill few requirement of industry & unable to fullfill few more requirement of industry. Hence Sun microsystem developers revised the concept of existing data structure & release to real Industry on the name of New collection framework and existing data structure is carry forwarded on the name of Legacy collection framework.
- In other words we have two types of collection framework; they are –
  1. New collection Framework
  2. Legacy collection Framework

1. New collection framework :-

Revised, from existing data structures of Java is known as New collection Framework.

New CFW is classified into two types : they are

(a) 1-D/single Collection Framework

(b) 2-D/Double collection Framework/ Maps

(a) 1-D/single collection Framework :-

- In this CFW Programming the data is organizing in the form of either row or column, but not in the form of rows and column or not in the form of key, values pairs.
- To deal with 1-D CFW programming we need to learn predefine Interfaces & corresponding pre-define classes. which are present in Java.util.* Package.

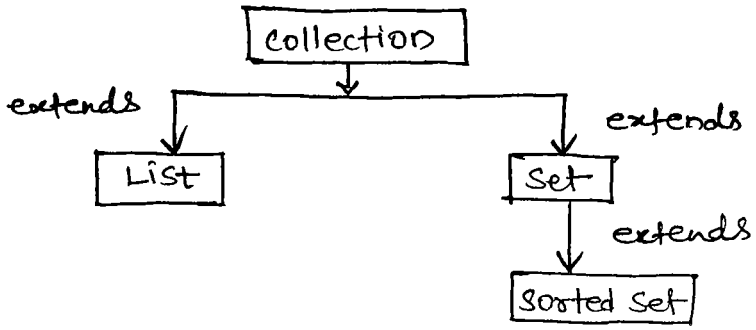
- 1-D collection Framework Interfaces

In new collection Framework we have four types of 1-D collection framework interfaces they are –

- Java.util.collection
- Java.util.List
- Java.util.Set
- Java.util.SortedSet

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Hierarchy chart



- The object of collection, List, Set, SortedSet Interface are known as 1-D collection Framework variable; and they allows us to organize the data either in the form of row or in the form of column but not in the form of row and column.

Differences between collection, List, Set, SortedSet :-

Differentiation factor	Collection	List	Set	SortedSet
Hierarchy	Top	List extends collection	Set extends collection	SortedSet extends set
functionality	Duplicates elements	Duplicates elements	unique ele.	unique ele.
Display	Random order	Sequential order	Random order	Sorted order
organizing difference	only at end	-Begining -Ending -middle	only at end	-Begining -Ending -middle
* Retrieval difference	• Forward direction ✓ • Backward direction ✗ • Random direction ✗	• Forward direction ✓ • Backward direction ✓ • Random direction ✓	• Forward direction ✓ • Backward direction ✗ • Random direction ✗	• Forward direction ✓ • Backward direction ✓ • Random direction ✓

Note :-

The objects of collections & set Interfaces are known as unit directional CFW object because they allows us to retreive the data only in forward direction.

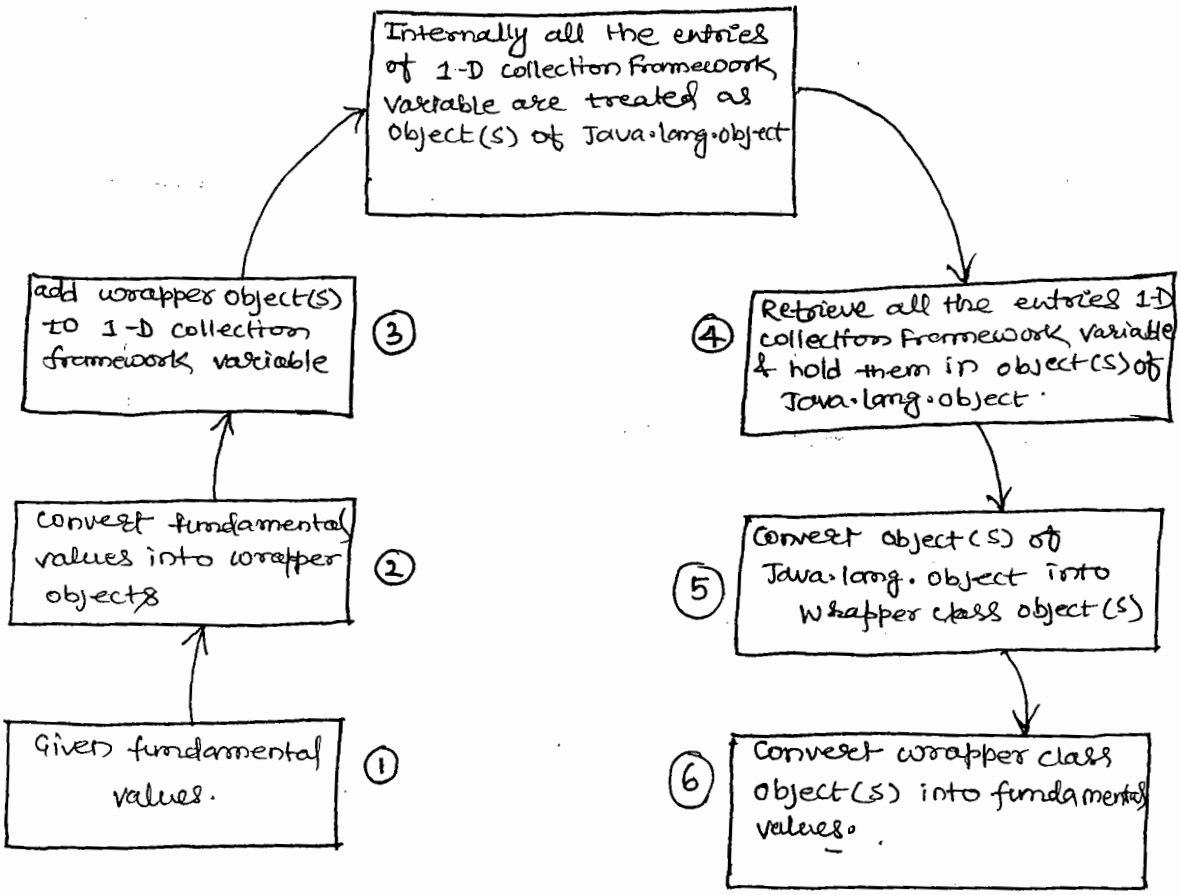
The objects of List & SortedSet are known as multidirectional CFW objects because they allows us to retrieve the data in all the direction.

1-D CFW process :-

1-D collection framework process contains 2 phases. They are-

(1.) Grouping phase
(2.) Un-grouping phase

The following diagram gives sequence of steps for performing grouping & un-grouping phases.



Here step ①, ②, ③ are used for grouping &
step ④, ⑤, ⑥ are used for un-grouping.

Object TypeCasting :-

casting is nothing but a process of converting one type of value to another type of value.

def:

The process of converting refrence of base class object into Subclass object refrence is known as object typecasting. (or class typecasting)

Syntax

```
SubClass Name  Subobj = (Subclass Name) Baseclass obj ;
```

need of object typecasting —

1. The result of the Java Program is available in the object of Java.lang.object.

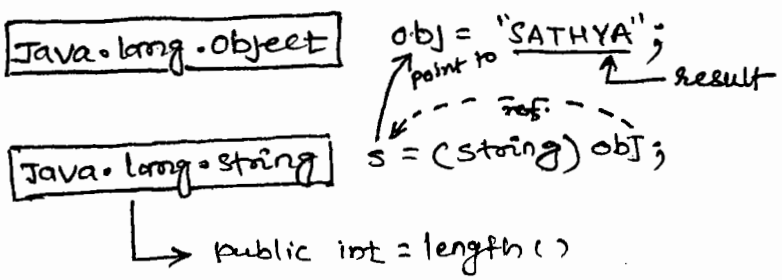
2. To process the result of object of Java.lang.object, there are no method present in Java.lang.object.

3. To process the result of object of Java.lang.Object, we have the special method present in ~~sub~~ class of Java.lang.object.

4. To process the result of object of Java.lang.object, w.r.t. object of Java.lang.object we can't access Special methods subclass of Java.lang.object. (according to Scope of base class object - see previous page - Inheritance)

5. To process the result of object of Java.lang.object, w.r.t. special methods of subclass of Java.lang.object we use object typecasting concept.

consider the following diagram -



ex:-

```
Object obj = "sathya";
sop("val of obj = "+obj); // sathya
int noc = obj.length(); //Error
String s = (string) obj; // object-type casting
int noc = s.length();
sop(noc); // 6
```

Q. write a Java program which illustrate the concept of fundamental variable type casting and object type casting.

```
// TypeCast.Java
Class TypeCast
{
 psvm (String [] args)
 {
 // Fund values type casting
    sop("Fund values type casting- operation");
    sop("_____");
    int a=10;
    float b;
    b=a; // implicit type casting
    sop("val of a(int) ="+a); // 10
    sop("val of a(float) ="+b); // 10.0
    float x = 10.75f;
    int y;
    y = (int)x; // explicit type casting
    sop("val of a(int) ="+a); // 10
    sop("val of b(float) ="+b); // 10.0
    float x = 10.75f;
    int y;
    y = (int)x; // explicit type casting
    sop("val of x(float) ="+x); // 10.75
    sop("val of y (int) ="+y); // 10
    int K=65;
    char l=(char) K; // explicit type casting
    sop("val of K(int) ="+K); // 65
    sop("val of l(char) ="+l) // A
```

```
Char p = 'a';
int q;
q = p; // Implicit type casting
sop(" val of p(char) =" + p); // 'a'
sop(" val of q(int) =" + q); // 97
sop(" ____________"); // object type casting
sop(" object type-casting operation");
sop(" ____________");
object obj = "SATHYA TECH";
sop(" val of obj =" + obj); // SATHYA
// int noc = obj.length(); ___ invalid
String s = (String) obj; // object type casting
int noc = s.length();
sop(" no. of chars =" + noc);
object obj1 = 100;
sop(" val of obj1 =" + obj1); // 100
// obj1 = obj1 + 1; ____ invalid
Integer io = (Integer) obj1; // object type casting
io = io + 1; // auto-unboxing
sop(" val of io =" + io); // 101
int x1 = io.intValue(); // prog-unboxing;
x1 = x1 + 2;
sop(" val of x1 =" + x1); // 103
sop(" ____________");
}
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

✱ methods in Java.util.Collection :-

1. **Public int size():-**

This method is used for finding the no. of entries present in 1-D CFW variable.

2. **Public boolean isEmpty():-**

This method returns true provided 1-D collection framework is empty otherwise it return False (non-empty).

3. **Public void clear():-**

This method is used for clearing all the elements of 1-D CFW variable.

4. **public boolean Contains(Object):-**

This method is used for finding the existence of an element in 1-D collection framework variable this method return true provided the specified element found in 1-D CFW variable otherwise it return false.

5. **Public boolean add (object):-**

This method is used for adding the elements to any CFW variable. This methods returns true provided as long as we call w.r.t. objects of collections of list and also returns true provided we add unique elements to Second sorted set object. This method return false; provided when we add duplicate element to second sorted set.

upto JDK 1.4:-

```
int a=10;
Integer io = new integer(0);
c.add(io);
```

(OR)

```
c.add(new Integer(10));
```

From JDK 1.5 version onwards:-

```
c.add(10);
c.add(10.75f);
c.add('A');
c.add(true);
SOP(c); // [10, 10.75, A, true]
```

c ← 1-D CFW variable

10
10.75
A
True

Note:-

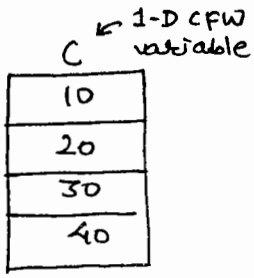
Here, add method is performing grouping phase.

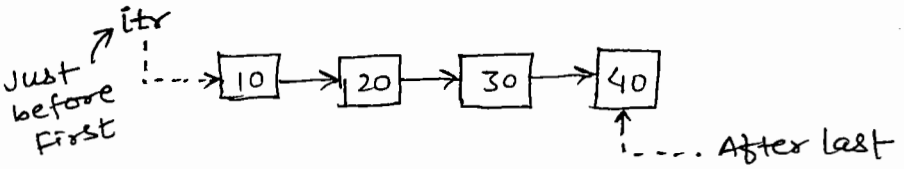
6. **Public Iterator iterator() :-**

This method is used for retreiving the data from 1-D CFW variable and forms forward directional chain. This method returns an object of iterator interface and it is just pointing to before the first element of forward directional chain.

ex:-

```
sop(c); // [10, 20, 30, 40]
int s=0;
Integrator itr = c.iterator();
```





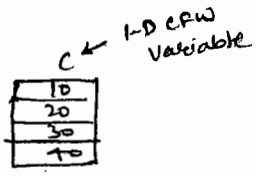

```
while (itr.hasNext())
{
  Object obj = itr.next(); // 1-D CFW process no-(4)
      Iterator io = (Iterator) obj; // (5)
        int x = io.intValue(); // (6)
          s=s+x;
          Sop(x); // 10, 20, 30, 40
}
Sop ("sum:"+s);
```

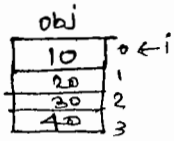
7. **Public Object[] toArray[] :-**

This method is used for retreiving all the elements of 1-D CFW-variable and converted into Array of Objects of Java.lang.Object.

ex:-

```
int s=0;
sop(c); // [10, 20, 30, 40]
Object obj[] = c.toArray(); // (4)
for (int i=0; i<obj.length; i++)
{
  Integer io = (Integer) obj[i]; // (5)
  int x = io.intValue(); // (6)
```





```
        s = s + x;
        SOP(x);   // 10, 20, 30, 40
    }
```

* Iterator :-

- It is one of the predefine interface present in Java.util.* Package.
- The purpose of interator interface object is that to retreive the data from 1-D collection framework variable in forward direction but not in backward dir^n.
- An object of iterator is by default pointing just before the first element of any collection framework variable.

Methods in Iterator :-

① public boolean hasNext()
② public object next()

- method ① returns 'true' provided an object of iterator is having next element otherwise it returns 'false'.
- method ② is used for retreiving the next element of any coll^n f.w. variable w.r.t. object of iterator.

→ Methods in Java.util.List

- we know that List is sub-interface of collection so that all the methods of collection are inherited into List.
- List contains the following special methods -

→ public void add (int, object);

This method is used for adding an element to list ~~object~~ interface obj either in begining, middle or ending.

ex:- Insert "sathya" in begining

```
l.add (0, "sathya")
```

- Insert an element sathya in the last position -

```
l.add (l.size(), "sathya");
```

represent last position where the element it above to insert.

- Insert the element "sathya" in the middle of the existing ele-

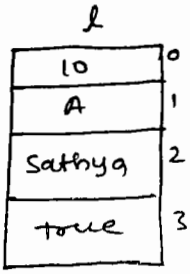
```
l.add (3, "sathya");
```

- Public Object get(int)

This method is used for retreiving an ~~object~~ element from List object by passing valid existing Position otherwise we get java.lang.IndexOutOfBoundException.

ex:-

```
Object obj = l.get(2);   // valid position
sop(obj); // sathya
```



ex:-

```
Object obj = l.get(30);   // invalid position
```

obj → null

l.get(30) → throws → IndexOutOfBoundException

- public void remove(object) :-

This method is used for removing an element of list object by passing its content.

- public void removeAll()

This method is used for removing all elements of list objects.

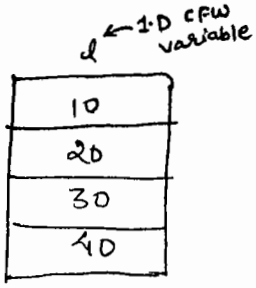
***

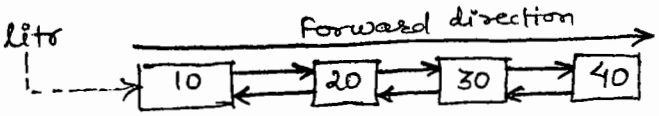
- Public ListIterator ListIterator()

- This method is used for retreiving the data from any C.F. variable and forms bidirectional chain.
- This methods returns an object of ListIterator which is by default pointing just before of first element of bidirectional chain.

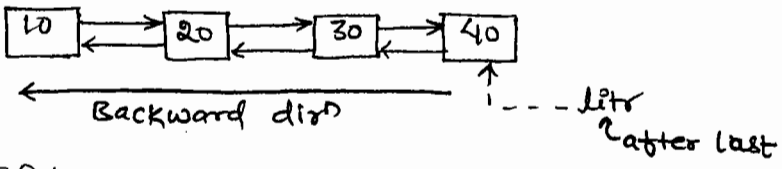
ex:-

```
sop(l); // [10, 20, 30, 40]
ListIterator litr = l.listIterator();
```






```
while (litr.hasNext())
{
    object obj = litr.next();
    sop(obj);   // 10 20 30 40
}
```




```
int s=0;
while (list.hasPrevious())
  {
    object obj = litr.previous(); // 4. 1-D CFW Phase
    Integer io = (Integer) obj; // 5 - 1-D-CFW phase
    int x = io.intValue(); // 6 - 1D CFW phase
        s=s+x;
        sop(x); // 40 30 20 10
  }
    sop("sum =" +s); // 100
```

List Iterator –

- It is one of the predefine sub-interface of iterator interface & present in Java.util.*
- The purpose of list Iterator interface object is to retreive the data from any C.F.W variable in both Forward & backward direction.
- Like IteratorInterface object ListIterator interface object is also by default pointing just before the first element of bidirectional chain.

method:–

(1) Public boolean hasNext() } Inherited from Iterator.
(2) Public Object next()
(3) Public boolean hasPrevious()
(4) Public boolean Previous()

- method (3) Provided 'TRUE" when List Interface object is having previous element otherwise it return in "False".
- method (4) is used for obtaining previous elements w.r.t. ListIterator Interface object.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. what are the different method used, for retrieving from 1-D C.F.W variable?

- we have four method to retreive the data from any 1-D CFW variable –

① public Iterator Iterator()
② Public Object[] toArray()
③ Public Object get(int)
④ Public ListIterator ListIterator()

– Methods in Java.util.set :-

- we know that set is the subinterface of collection, so that all the methods of collection are inherited into set interface.

"Set Interface does not contain any special method except the method of collection".

- Even though the method of collection & set Interfaces is same; Collection methods defined in some predefined class in such a way that duplicate elements are allowed. where as set method are defined in some other predefined class in such a way that unique elements are allowed.

- In otherwords eventhough the method of collection & set Interface are same, their implementation are different in different predefined classes which will satisfy collection & set properties.

- methods in Java.util.sortedSet :-

- We know that sortedSet is sub interface of set, so that all the methods of set are inherited into sortedSet and it contains some special following method :—

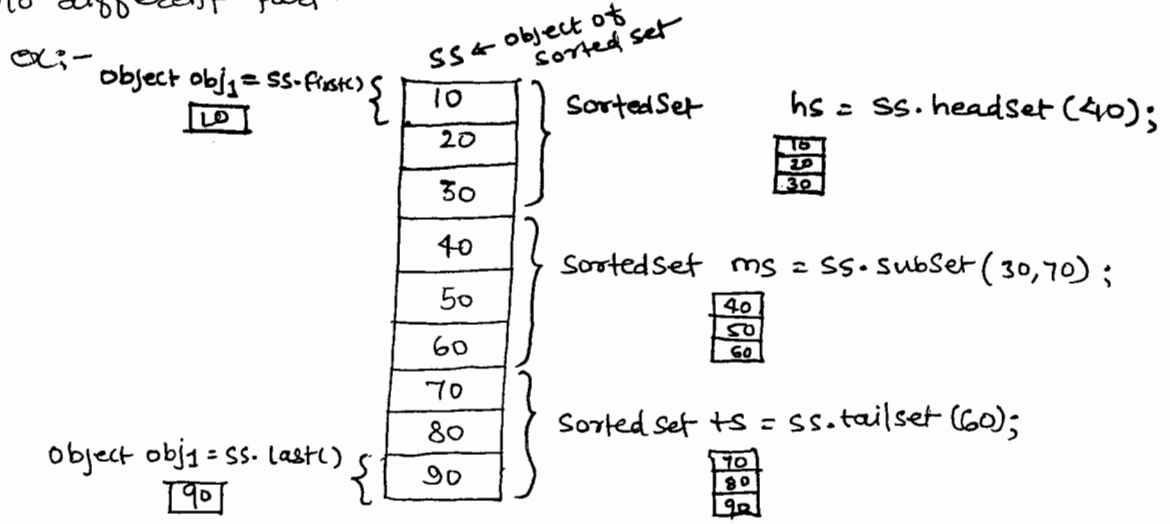
1. – Public Object first()
2. – public Object last()
3. – public SortedSet headSet(Object obj) ← target
   ↳ $x_i < obj$
4. – Public SortedSet subSet(Object obj1, Object obj2)
   ↳ $obj_1 < x_i < obj_2$
5. – Public SortedSet tailSet(Object obj)
   ↳ $x_i > obj$

- method ① & ② are used for obtaining 1st & Last element of SortedSet.
- method ③ is used for obtaining few top most value (x_i) which is less than target object. $\boxed{x_i < obj}$
- method ④ is used for obtaining few middle value (x_i) which are greater than the target object (obj_1) and target object (obj_2). i.e. $\boxed{obj_1 < x_i < obj_2}$

- method ⑤ used for few bottom most value (x_i) which are greater than target object obj mathamatically $\boxed{x_i > obj}$.

All the above method are used for spilitting the sorted set elements into different part.

ex:-



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

1-D collection frame Work Classes :—

- 1-D CFW classes contains definition for the abstract methods which are inherited from 1-D collection Framework interfaces.
- The following table gives 1-D collection framework interface name, corresponding 1-D CFW class name & there hierarchy :—

1-D CFW Interface Name	1-D CFW class Name	Hierarchy
① collection	① AbstractCollection	← Impliments — collection
② List	② AbstractList	← extends — AbstractCollection; ← impliments — List
③ Set	③ AbstractSet	← extends — AbstractCollection; ← impliments — Set
④ SortedSet		
	④ AbstractSequentialList	← extends — AbstractList
	⑤ LinkedList, ⑥ ArrayList	← extends — AbstractSequentialList
	⑦ HashSet	← extends — Abstract Set
	⑧ TreeSet	← extends — AbstractSet; ← implements — SortedSet.

- In above table the classes ①, ②, ③, ④ are the predefine 1-D CFW Abstract classes and they are reserved by SUN-microsystem for the future innovation in 1-D CFW Prog. and they made as Abstract. so that we can not create their object directly. The classes ⑤, ⑥, ⑦, ⑧ are the bottom most concrete subclass in 1-D CFW and they full fill in The current industry requirement in 1-D CFW.

Note —
CFW contains three types of layers they are —

ⓐ Interfaces layer

ⓑ Abstract classes layer

ⓒ Concrete classes layer.

LinkedList :-

- LinkedList is a concept of data structures.

☆ adv. of linkedList :-

If we develop any Java application with linkedList concept we get the following advantages:-

- It organizes both homogeneous & heterogeneous elements.
- It provides dynamic size (expandable)
- It provides serializability.
- It provides dynamic insertion facility (No overlapping problem).

- To get the above advantage of linkedList in Java Programming are have a predefine class called java.util.linkedList.

- Programmatically to use linkedList to our program we must create an object of linkdList class.

```
LinkedList ll = new LinkedList ();
```

Here; ll is called 1-D CFW variable.

- In linked list the data is organizing in the form of "Nodes".
- Node contains two parts and whose str is given below —

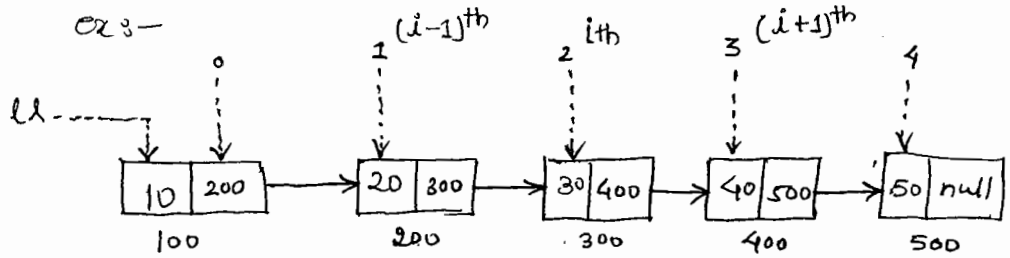
Node

Data Part	Address Part

- Here Node represents object of linked list.
- Data part represents type of data being stored.
- Address part represents address of next node.

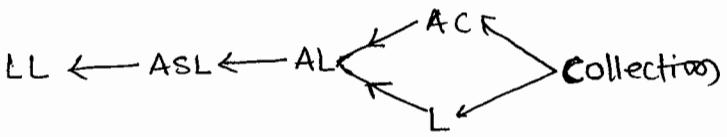
An address part of last node contains null which indicates end of linkedList.

Ex:-



The data organizing principle followed in linkedList is that Address of ith node stored in Address Part of (i-1)th node, Address of (i+1)th node stored in the Address part of last node contains null which indicate end of linkedList.

The Hierarchy of LinkedList is shown below:—



LL ⇒ Linked List
ASL ⇒ Abstract Subsequent List.
AL ⇒ AbstractList
AC ⇒ AbstractCollection
L ⇒ List

Profile of Java.util.LinkedList:-

Constructor

1. LinkedList() :-
2. LinkedList (int);

Instance method :-

3. Public void addFirst (object)
4. Public void addLast (object)
5. Public object getFirst ();
6. Public object getLast ();
7. Public object removeFirst ()
8. Public object removeLast ()

- Constructor 1 is used for creating an object of LinkedList.
  ex:- linkedlist ll =
- constructor 2 is used for creating an object of linkedList by specifying no. of initial nodes.
  LinkedList ll = new LinkedList (4);
- methods 5 & 6 are used for retreiving first and Last elements of linked list.

ex:-
```
Object obj1 = ll.getFirst ();
        sop (obj1);
        (OR)
Object obj1 = ll.get (0);
```
ref. to first ele. position

```
object obj2 = ll.getLast ();
        sop (obj2)
        (OR)
Object obj2 = ll.get (ll.size () - 1);
        sop (obj2)
```
↳ last element position.

- methods ③ & ④ are used for adding the element to the linkedList either in begining or the last.

ex:-

- ll.addFirst(10);

  (or)

  ll.add(0,10); ← first position

- ll.addLast(50);

  (OR)

  ll.add(ll.size(),50); ← last element position where an element inserting.

- methods ⑦ & ⑧ removing first & last element of linked list.

ex:- Object obj1 = ll.removeFirst();

(or)

Object obj1 = ll.remove(ll.getFirst(0));

Object obj2 = ll.removeLast();

(or)

Object obj2 = ll.remove(ll.get(ll.size()-1))

Q. write a java program which illustrate the concept of Linked List.

```
// L.java
import java.util.*;
class LL
{
 psv main (String[] args)
 {
   LinkedList ll = new LinkedList();
   sop ("content of ll =" +ll); // []
   sop ("size of ll = " +ll.size()); //0
// add the data to ll - 1D CFW var
      ll.add (20);
      ll.add (30);
      ll.add (40);
```

```
sop ("content of ll =" + ll); // []
sop ("size of ll =" + ll.size()); //5
// add the elements 10 and 50 at first & last position
    ll.addFirst(10);
    ll.addLast(50);
sop ("content of ll =" + ll); // [10, 20, 30, 40, 50]
sop ("size of ll =" + ll.size()); //7
// extract the data from ll
sop ("extract the data from ll using iterator()");
sop ("————————");
int s=0;
iterator itr = ll.iterator();
while (itr.hasNext())
{
    Object obj = itr.next(); // step-4
    Integer io = (integer)obj; // step-5
    int x = io.intValue(); // step-6
    sop(x);
    s=s+x;
} // while
sop ("Sum using iterator() =" + s); //100
sop ("————————");
sop ("extract the data from ll using Iterator()-FD");
sop ("————————");
ListIterator litr = ll.listIterator();
while (litr.hasNext())
{
    Object obj = litr.next();
    sop (obj);
}
sop ("————————");
sop ("extract the data from ll using listIterator()-BD");
sop ("————————");
```

```
int s1 = 0;
while (litr.hasprevious())
 {
   object obj = litr.previous();
   Integer io = (Integer) obj ;
   int x = io.intValue();
    s1 = s1+x;
    sop (obj);
 }
sop ("sum using ListIterator() ="+s1); // 100
sop ("_____");
 sop (" extract the data from ll using toArray()");
 sop ("_____");
  Object obj[] = ll.toArray();
  int s2=0;
   for (int i=0; i<obj.length ; i++)
    {
      integer io = (integer) obj[i];
      int x = io.intValue();
       sop (x);
       s2 = s2+x;
    }
  sop ("sum using to Array () =" +s2); // 100
  sop ("_____");
  // extract the random element
       Object robj = ll.get(2);
       sop ("random elements = " robj); //30
    } // main()
} // LL-class.
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. write a Java Program which will accept three integer value & find the biggest element return all three source value along with biggest value.

```
// Big.Java
  Package bP;
  import Java.util.*;
  public class Big
  {
   int a, b, c;
   Public void readValues()
   {
    Scanner s = new Scanner(System.in);
    System.out.Println("Enter First values");
    a = Integer.parseInt(s.nextLine());
    SOP("Enter second value");
    b = Integer.parseInt(s.nextLine());
    SOP("Enter third value:");
    c = Integer.parseInt(s.nextLine());
   }
    public LinkedList FindBig()
    {
      int big = a;
      if (b > big)
      {
        big = b;
      }
      if (c > big)
      {
        big = c;
      }
// return all 3 source values and big values with CFW var
      LinkedList ll = new LinkedList();
        ll.add("val of a=" + a);
        ll.add("val of b=" + b);
        ll.add("val of c=" + c);
        ll.add("biggest =" + big);
        return ll;
    }
```

```
public void disp(LinkedList ll)
{
  Iterator itr = ll.iterator();
  sop("_____");
  sop("result");
  sop("_____");
  while (itr.hasNext())
  {
    object obj = itr.next();
    sop(obj);
  }
  sop("_____");
}
}
```

compile the above Prog—

```
javac -d Big.java
```

bp → Big.class

Q write a java main program which makes use of bp.Big class.

```
// BigDemo.java
import java.util.*;
import bp.Big;
class BigDemo
{
  psvm(String[] args)
  {
    Big bo = new Big();
    bo.readValues();
    LinkedList ll = bo.findBig();
        bo.Big(ll);
  }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

* Limitation of LinkedList :-

1. LinkedList based Appl^n takes more memory space.
2. LinkedList based Appl^n takes more retreival time & it reflects on execution time.
3. overall performance of LL based Appl^n is less.

To overcome the limitation of LinkedList we have an another predefine class called Java.util.ArrayList

Note:-

- Anything we store in heap memory takes more amount of space & anything we retreive from Heap memory takes more retrieval time.
- where as anything we retreive from Associative memory it takes least amount of retrieval time.
- The linked list store data Part and address part in Heap memory but in Arraylist it store data Part in heap & Address Part in Associative memory; that why the retrieval time of Arraylist is more; and performance is better.

Java.util.ArrayList

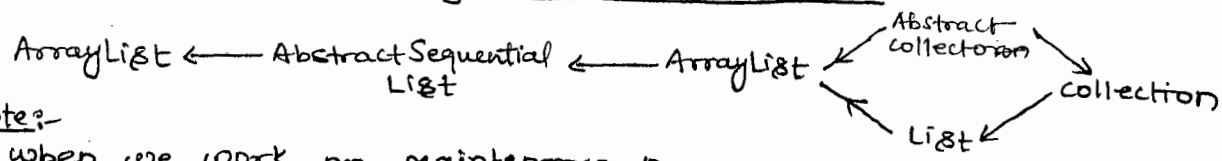
- It is the one of the bottom most concrete class in 1-D CFW.
- The internally implementation of arraylist class is associative memory

- advantage of arraylist over LinkedList:-
  - ArrayList based appl^n takes less memory space.
  - ArrayList based appl^n takes less retrieval time. [Address Part is stored in associated memory]
  - The overall performance of arraylist is more.

- In arraylist data is organizing in the form of ~~cells~~ cells.
- cell values are stored in Heap memory & cell address are stored in associative memory.
- creating an arraylist is nothing but creating an obj of arrayList class.

```
ArrayList al = new ArrayList();
```

The Hierarchy of arrayList is shown below —



Note:-

- when we work on maintenance project ~~of~~ somebody all ready used LinkedList then we must replace the linkedList with ArrayList for getting the advantage of ArrayList.

DTO - Data Transfer object is one which will ~~written~~ all the value at once irrespective of their types.

Q. write a Java program which illustrate the DTO & DAO ~~of~~ by using ArrayList.

DAO - A DAO is one which will be access all the value irrespective of their type. (As a Part of JEE a collection FrameWork Variable treated as DAO & DTO)

```
// Emp.java
Package ep;
Public class Emp.
{
 Public    ArrayList empDTO()
 {
  Scanner s = new Scanner (System.in);
   sop ("Enter Emp. Number");
   int eno = Integer.parseInt (s.nextline());
   sop( "Enter Emp name:");
String name = s.nextline ();
   sop ("Enter Emp sal:");
    Float sal = Float.ParseFloat ( s.nextLine());
     sop ("enter Emp Desig:");
      String desg = s.nextline();
//return all emp values — with - CRUD. var
        ArrayList al = new ArryList ();
         al.add ("Emp number =" +eno);
         al.add ("Emp name =" + name);
         al.add ("Emp sal =" + sal);
         al.add ("Emp Desig =" +desg);
         return al;
 }
 Public void dispEmpDAO (Arraylist al)
 {
     sop ("————");
     sop ("Emp Data");
```

```
sop("_____");
Object obj[] = xl.toArray();
for(object emp:obj) //enhanced model 'for' loop.
{
 sop(emp);
}
sop("_____");
}
}
```

compile the above program

```
javac -d Emp.java
```

Q. write a java program (main) ; which makes use of Emp.Java -

```
// DAODemo.Java
import ep.Emp;
import java.util.*;
class DAODemo
{
 psvm (String [] args)
 {
  Emp eo = new Emp();
  ArrayList al = eo.empDTO();
  eo.dispEmpDAO(al);
 }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

HashSet & TreeSet

Hash Set

① HashSet object never allows duplicates.

② HashSet class object organizes the data in the form of HashTable by following Hashing Mechanism.

③ we can't determine in which order HashSet class object displays the data, because Sun microsystem developers didn't disclose which type of Hashing Mechanism followed in the implementation of Hash Set class.

④ HashSet class object the operation like-
- Insert
- modify
- delete
- search / lookup

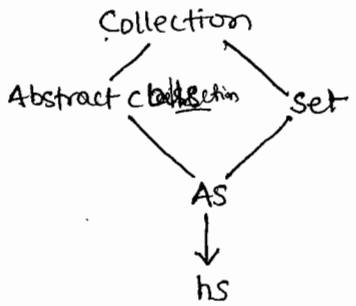
takes much time.

⑤ Retreiving the data from Hash-Set takes more amount of time.

⑥ creating HashSet is nothing but creating an obj of HashSet class.

```
HashSet hs = new HashSet();
```

⑦ Hierarchy –



TreeSet

Ⓐ TreeSet class object also never allows duplicates.

Ⓑ TreeSet class object organizes the data in the form of Nodes by following Binary trees Concepts.

Ⓒ TreeSet class object displays the data in sorted form.

Ⓓ All treeSet class object the operation like –
- Insert
- Delete
- search / Lookup
- modify

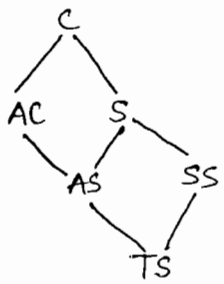
takes Less amount of time.

Ⓔ Retreiving the data from Tree set takes Less amount of time.

Ⓕ creating TreeSet is nothing but creating an obj of TreeSet class

```
TreeSet ts = new TreeSet();
```

Ⓖ - Hierarchy



Q. write a Java program which illustrate the concept of HashSet & TreeSet.

```
//hsts.java
import java.util.*;
Class hsts
{
 PSVM (String [] args)
 {
  //HashSet hs = new HashSet();
  TreeSet hs = new TreeSet();
  sop ("contents of hs =" +hs); // []
  sop (" size of hs =" +hs.size()); // 0
  // add the data to hs
     hs.add (28);
     hs.add (8);
     hs.add (30);
     hs.add (-100);
     hs.add (2);
     hs.add (28); // executed but not added
     sop ("contents of hs =" +hs); // [...]
     sop(" size of hs =" +hs.size()); // 5
     sop(" _____");
     sop(" extract the data from hs");
     sop (" _____");
     Iterator itr = hs.iterator();
     while (itr.hasNext());
     {
      object obj = itr.next();
      System.out.println(obj);
     }
     sop (" _____");
 }
}
```

2-D Collection - FrameWork

- In 2-D collection Framework the data is organizing in the form of (key, value). In (key, value) the value of key represents unique. and the value of value may or maynot be unique. i.e. duplicates can be allowed.

ex:-

Deposit

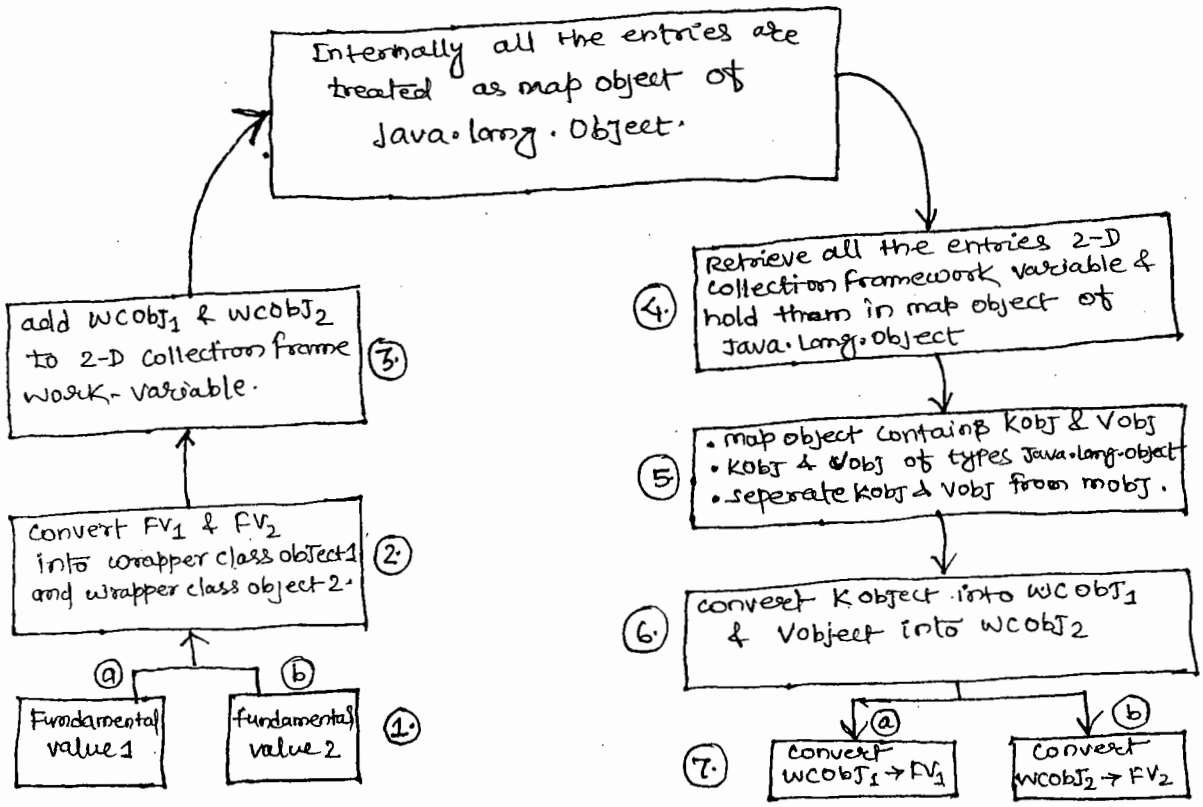
ac no.	bal.
10	1.5
20	2.5
30	1.5

Here; ac no ⇒ Name of key
bal ⇒ Name of value.
(10, 20, 30) ⇒ values of key
(1.5, 2.5, 1.5) ⇒ values of value.

2-D collection Framework Process :-

2-D collection Framework Process contains two phases. They are —
① Grouping/Assembling phase ② Un-grouping/Deassembling phase

- The following diagram gives sequence of steps for grouping & ungrouping phases —



- In the above diagram ①, ②, ③ steps are used for Grouping phase and ④, ⑤, ⑥, ⑦ are used for un-grouping phase.

- To deal with 2-D collection Framework Programming, we need to learn 2-D collection Framework Interfaces & corresponding classes which are present in Java.util.*;

2-D/double collection Framework (or) map interfaces :-

2-D interfaces are of three types. They are —

1. Java.util.Map
2. Java.util.map.Entry
3. Java.util.SortedMap.

1. Java.util.Map :-

- It is one of the top-most predefine interface in 2-D CFW interfaces.
- An object of Map interface organizes the data in the form of (key, value) and in which values of key are unique and values of value may or maynot be unique i.e. duplicates are allowed.
- Map interface object displays the data in random order. In other words whatever the order we add the data to map interface object in the same order the data will be displays.

methods in Java.util.map :-

1.) Public int size()
This method is used for finding the no. of entries present in any 2-D collection framework variable.

2.) Public boolean isEmpty() :-
This method returns true provided 2-D collection framework ~~for~~ variable is empty otherwise it returns false.

3.) Public void put (object, object) :-
This method is used for inserting (key, value) in 2-D CFW variable and it can be also used for modifying the existing values of value by passing existing key.

ex:-

m ← 2-D collection F.W. variable

```
m.put(10, 1.5f);
m.put(20, 2.5f);
m.put(30, 1.5f);
```

Inserted entries ⟹

10	1.5
20	2.5
30	1.5

```
sop(m); //{(10=1.5), (20=2.5), (30=1.5)}
```

```
m.put(10, 9.5)
```

Modified entries ⟹

m ← 2-D collection F.W. variable

10	9.5
20	2.5
30	1.5

```
sop(m); //{(10=9.5), (20=2.5), (30=1.5)}
```

4. <u>Public Object get(object)</u> :-

This method is used for obtaining the values of value by passing value of key.

```
ex:- Object bal = m.get(10);
     [1.5]

     sop(bal); // 1.5

     Object bal1 = m.get(100);
     [null]

     sop(bal1); // null
```

5. <u>Public Set keySet()</u> :-

This method is used for obtaining set of keys from any 2-D collection framework variable and hold them in the object of Java.util.Set interface.

ex:-

```
sop(m); // {(10=1.5), (20=2.5), (30=1.5)};
Set s = m.keySet();
  // s = [10, 20, 30]
Iterator itr = s.iterator();
```

m ← 2-D collection framework variable.

10	1.5
20	2.5
30	1.5

itr → [10] → [20] → [30]

```
while (itr.hasNext())
{
  Object kobj = itr.next();
  Object vobj = m.get(kobj);
  Integer io = (Integer) kobj;
  float fo = (float) vobj;
```

```
int acno = io.intValue();
float bal = fo.floatValue();
sop (bal + " is bal of" + acno);
}
```

* 6. <u>public Set entrySet()</u>

This method is used for retrieving all the entries of 2-D collection framework variable and hold them in the object of Java.util.Set interface.

ex:-

```
sop (m); // {(10=1.5), (20=2.5), (30=1.5)}
Set s = m.entrySet();
// s = [(10=1.5), (20=2.5), (30=1.5)]
Iterator itr = s.iterator();
```

itr → [10=1.5] → [20=2.5] → [30=1.5]

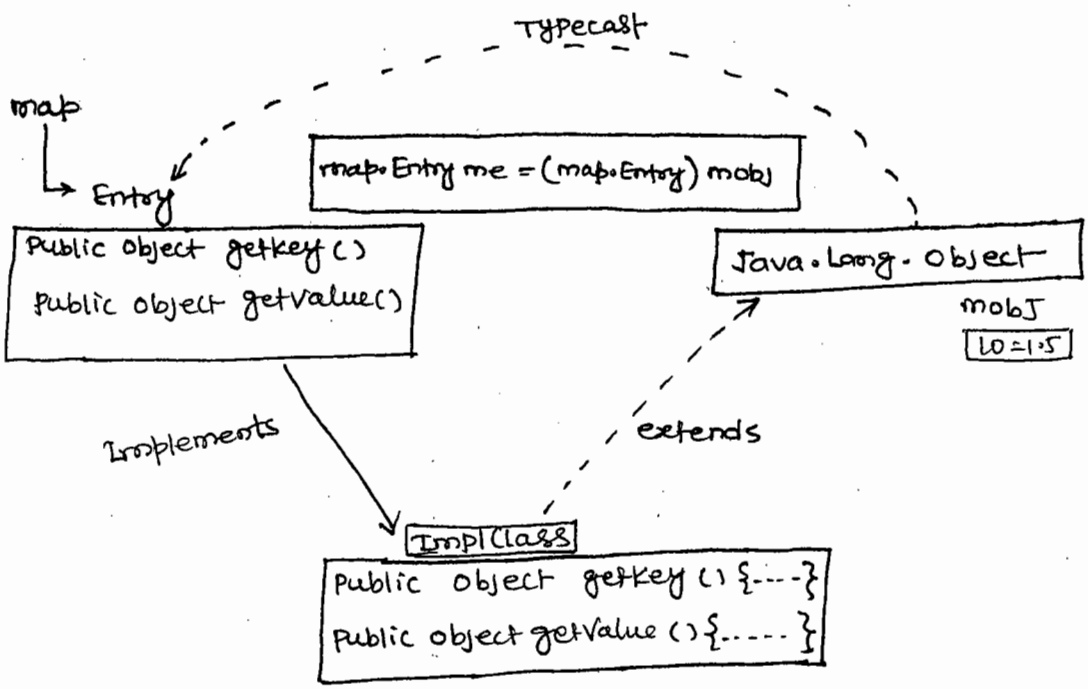
```
while (itr.hasNext())
{
  Object mobj = itr.next(); // (4)
        [10=1.5]
  map.Entry me = (map.Entry) mobj;
  Object kobj = me.getkey();
  Object vobj = me.getvalue(); // (5)
  Integer io = (Integer) kobj; // (6)
  float fo = (float) vobj;
  int acno = io.intValue();
  float bal = fo.floatValue(); // (7)
```

<u>Note</u>:-

As a part of Java programming one can typecaste an object of Java.lang.object into any of its subclass object but we can also typecast an obj. of Java.Lang.object into an object of Interface.

<u>Formula for Object of Java.lang.Object Typecasting into interface object</u>:-

An object of Java.lang.object which is a super class of implementation class is typecasted to an object of SuperInterface of implementation class which is a sub-class of Java.lang.object.



2. **Java.util.map.Entry**

In map.Entry map is one of the outer interface & Entry is one of the inner-interface/nested interface.

- The purpose of map.Entry is to seperate key & value from mapObject.

methods:-

1. Public object get Key()
2. public object getValue()

The above two methods are used for seperating (key, value) from map object.

3. **Java.util.SortedMap**

Sorted map is a subinterface of map interface, so that all the methods of map are inherited into SortedMap.

An object of sorted map also organizes the data in the form of (key, value).

- Sorted map interface object displays the data in sorted order based on the value of key.

methods:-

1. Public object first();
2. Public object last();
3. Public SortedMap headmap(object kobJ)
   → $x_i < kobJ$

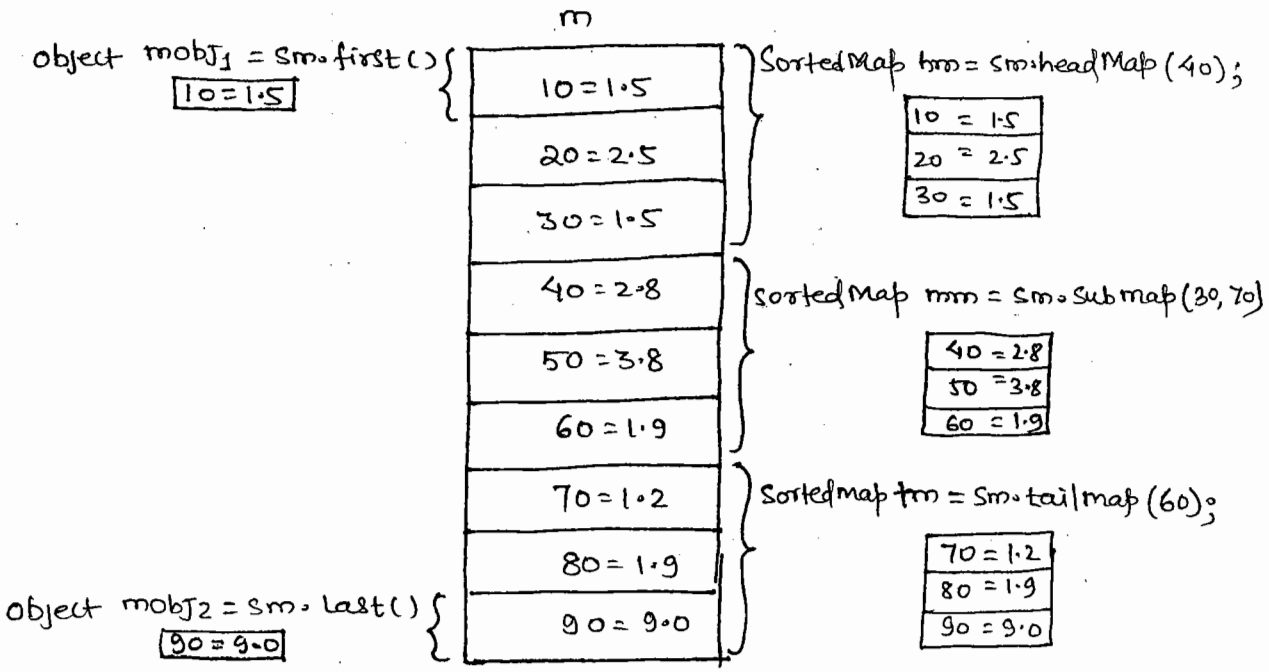
4. public SortedMap subMap (object KobJ1, object KobJ2)

↳ $\boxed{KobJ_1 < x_i < KobJ_2}$

5. public SortedMap tailMap (object KobJ)

↳ $\boxed{x_i > KobJ}$

for more explanation see the method of Java.util.SortedSet [Notes-1]



2-D Collection FrameWork Classes :-

- These classes contains definition for the abstract method of 2-D CFW interfaces.
- The following table gives 2-D CFW interface name, corresponding classname & their hierarchy.

Hierarchy chart :-

2-D C.F.W. Interface Name	2-D CFW Class Name	Hierarchy
1. Map	1. AbstractMap	← implements — Map
2. map.Entry	2. (a) HashMap $ Entry	← impl. — map.Entry
	(b) TreeMap $ Entry	← impl. — map.Entry
3. SortedMap	3. AbstractSortedMap	← extends — Abstract Map; ← Impl. — Sorted Map
	4. HashMap	← extends — AbstractMap
	5. TreeMap	← extends — Abstract SortedMap

- In the above table the classes (1) & (3) are belong to 2-D CFW abstract classes and they are reserved by SUN MICROSYSTEM developers for future innovation in 2-D CFW & there is no direct uses of these class in our real world application.
- The classes (4) & (5) represents Bottom-most Concrete sub-classes in 2-D collection-framework and we use in real-world-appl^n.
- In (2-a) & (2-b) Entry is the inner class of HashMap & TreeMap which is overridding the method of map.Entry interface for displaying the data in undetermine for & its sorted form respectively.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Hashmap & TreeMap

Hashmap

1. - An object of Hashmap allows to organize unique value for keys.

2. - An object of Hashmap allows us to organize the data in the form of (key, value) by following Hashing mechanism.

3. - we can't determine in which order the Hashmap class object displays the data i.e. un-determ-ind form.

4. - on Hashmap object the operations like insertion, deletion, modification and searching takes more time.

5. - retreiving the data from Hashmap takes more time.

6. - creating Hashmap is nothing but creating an object of Hashmap class.
   ex- Hashmap Hm = new Hashmap();

7. - Hierarchy -

   map
   ↓
   Abstractmap
   ↓
   Hashmap.

Treemap

- An object of Treemap allows unique values for keys.

- Treemap class object organizes the data in the form of (K, V) by following binary-tree's concept.

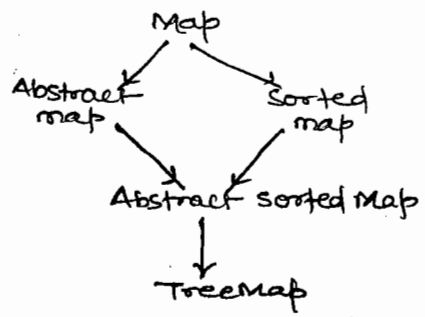
- Treemap class object always displays the data in sorted order (sorting can be done based on key)

- on treemap object the operation like deletion, insertion, searching, modification takes less time.

- retreiving the data from Treemap takes less time.

- creating Treemap is nothing but creating an object of Treemap class.
  ex -
  Treemap tm = new Treemap();

- Hierarchy -



Q. write a Java Program which illustrate the concept of HashMap and Treemap.

```
// hmtm.java
import java.util.*;
class hmtm
{
 P.S.V.M ( String [] args)
 {
//HashMap hm = new HashMap();
  TreeMap hm = new Treemap ();
   sop ("content of hm". + hm); // { }
    sop ( "size of hm ="+hm.size()); // 0
// add the data to hm
      hm.put (100, 1.5f);
      hm.put ( 10, 2.5f);
      hm.put (1, 5.5f);
      hm.put (90, 1.5f);
      hm.put (90, 3.5f); //executed and considered as modified
                           entry.
     sop ("content of hm="+hm); // {....}
     sop ("size of hm ="+hm.size()); // 4
// Extract the data from hm
      sop ("Extract the data from hm- entryset ()");
      Set s = hm.entryset ();
      Iterator itr = s.iterator();
      sop ("_____");
      while ( itr.hasNext())
      {
        Object mobj = itr.next(); // step-4
        map.Entry me = (map.Entry) mobj;
        Object Kobj = me.getkey();
        Object vobj = me.getValue();
        Integer io = (Integer) Kobj;
        Float fo = (Float)vobj;
        int acno = io.intValue();
        float bal = fo.floatValue();
        sop (bal+" is the bal of"+acno);
      } //while
```

```
sop ("————————");
sop ("Extract the data from hm-keyset()");
sop ("————————");
Set S1 = hm.keySet();
Iterator itr1 = S1.iterator();
while (itr1.hasNext())
{
  object kobj = itr1.next();
  Object vobj = hm.get(kobj);
  Integer io = (Integer) kobj;
  Float fo = (Float) vobj;
  int acno = io.intValue();
  float bal = fo.floatValue();
  sop ( bal + "——→" + acno);
}
sop ("————————");
} // Main
} // hmtm - ELC
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Legacy Collection FrameWork

- Renamed formed of existing Data Structures of Java is known as Legacy collection Framework.

☆ Differences between Legacy CFW & New CFW

Legacy CFW	New CFW
• Renamed form of existing Data Structure of Java is known as Legacy CFW.	- Revised form of existing Data structure of Java is known as New CFW.
• All the methods of Legacy CFW belongs to synchronized	- All the methods of N.CFW belongs to Non-synchronized.
- Legacy CFW variable gives Thread-safety result.	- New CFW variable gives Non-thread safety result.
- Legacy CFW API is highly recommended to use in client side / Non-server side applications.	• New collection framework API is recommended to use in server side application because server software provides synchronization facity for getting thread safety result.
- Like New CFW, Legacy CFW also contains 1-D & 2-D with corresponding interface & classes which are present in Java.util.*;	

- The following are the Legacy CFW 1-D & 2-D related interfaces and classes.

1.) Enumeration
2.) vector
3.) Stack
4.) Dictionary
5.) Hashtable.
6.) Properties.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

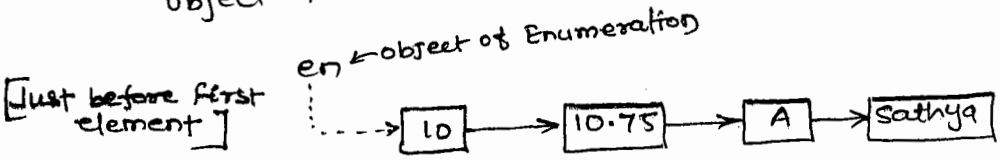
1. Enumeration –

- It is one of the predefine Interface present in Java.util.*;
- The purpose of Enumeration interface object is to extract the data from any Legacy CollectionFrame work variable in forward direction only.
- By default an object of Enumeration is pointing just before the first element of any Legacy collection frame work variable.
- The functionality of Enumeration is Less or more similar to iterator interface, but Enumeration object belongs to synchronized & iterator interface objects belongs to Non-Synchronized.

method –

1. public boolean hasMoreElements()
2. public Object nextElements()

- method 1 returns true provided Enumeration interface object having more element otherwise returns false.
- method 2 is used for obtaining next element of any Legacy CFW variable with respect to Enumeration interface object provided method 1 must return true.




```
while (en.hasMoreElement())
{
    object obj = en.nextElement();
        Sop(obj);

class c = obj.getClass();
```

→ class Java.Long.Integer

```
String name = c.getName();
```

→ Java.Long.Integer

```
if (Cname.equals("java.lang.Integer"))
{
  Integer io = (Integer)obj;   // DOTC
  int x = io.intValue();
}
if (Cname.equals("java.lang.Float"))
{
  Float fo = (Float)obj;   // Dynamic object Typecasting
  float y = fo.floatValue();
}
  . . . . . .
  . . . . . .
} //while
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Note :-

- Depends on the type of value present in the object of Java.lang.object. we are typecasting it into its equivalent sub class object is called "Dynamic object Type casting."

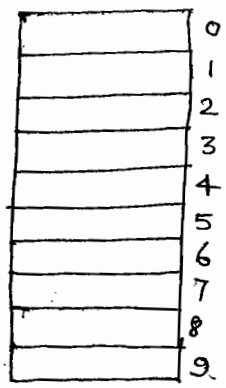
2. Vector :-

- Vector is one of the predefine 1-D Legacy CFW Concrete class.
- Vector class object organizes the data in the form of cells.
- Cell values are stored in Heap memory and cell addresses are stored in Associative Memory.
- The functionality of vector is similar to ArrayList but vector class object belongs to synchronized and ArrayList class object belongs to Non-Sync.
- Creating a vector is nothing but creating an object of vector class.

ex:-

```
Vector V = new Vector();
```

- The default capacity of the vector = 10 cells
  (No. of cells to be created initially)
- The default size of vector is 0 (zero).
  (The No. of values available by default in vector)



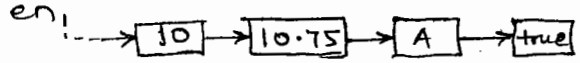
Profiles of vector:-

- Constructors
  1. vector ()
  2. vector (int)
- Instance Method
  3. Public int capacity ()
  4. Public int size()
  5. Public void addElement (objects)
  6. Public void addElement (int, object)
  7. public void insertElement (int, object)
  8. Public object getElement (int)
  9. public void setElement (int, object)
  10. public void removeElement (object)
  11. Public void removeElementAt (int)
  12. Public Enumeration element ()

- method (12) is used for retreiving all the elements of vectors and forms founded directional change.
  This method returns an objects of Enumeration which is by default pointing just before the first elements.

ex:- SOP (V); // [10, 10.75, A, true]

Enumeration en = v.elements();



10
10.75
A
True

Q. WAJP which illustrate vector & perform dynamic object typecasting.

```
// vect.java
import java.util.*;
class Vect
{
  PSVM (String [] args)
  {
    Vector v = new Vector();
    SOP ("content of v" +v); // [ ]
    SOP (" cap of v =" +v.capacity()); // 10
```

```
SOP ("size of v =" + v.size()); // 0
// add the data to v
    v.addElement (10);
    v.addElement (10.75);
    v.addElement ('A');
    v.addElement ("sathya");
    v.addElement (20);
    v.addElement (20.25);
    v.addElement ("tech");
    SOP ("content of v =" + v); // [10, 10.75, A, sathya]
    SOP ("cap of v =" + v.capacity());  // 10
    SOP ("size of v =" + v.size()); // 4
// extract the data from v
    int is = 0;
    double ds = 0.0;
    String s = " ";
    SOP ("Elements of v");
    SOP ("_________");
    Enumeration en = v.element();
    while (en.hasMoreElements())
    {
        object obj = en.nextElement();
        System.out.println (obj);
        Class c = obj.getClass();
        String Cname = c.getName();
        if (Cname.equals ("java.lang.Integer"))
        {
            Integer io = (Integer) obj; // dynamic obj. Type casting
            int x = io.intValue();
            is = is + x;
        }
        if (Cname.equals ("java.lang.Double"))
        {
            Double do1 = (Double) obj;
            double x = do1.doubleValue();
            ds = ds + x;
        }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
if (Cname . equals ("java.lang.String"))
 {
   String io = (String) obj ;
      s = s+io;
   }
 } // while
  sop ("————————");
  sop ("int sum =" +is);
   sop ("double sum =" +ds);
   sop ("String concat =" +s);
    sop ("————————");
  }
}
```

3. Stack :-

Stack is one of the concept of data structure and it can be implemented in many no. of languages.

Real-time implementation of Stack -

1. Converting Infix expression into either Postfix or prefix expression.
2. Evaluation of Prefix or Postfix expression.
3. Evaluation of ordinary method calls or Constructor calls.
4. Evaluation of Recursive function calls.

- The internal working principal of stack is LIFO
  LIFO says " The element which is inserted last is that element will be taken first".

  In the Java Prog. the concept of stack is given in the form of Pre-define class

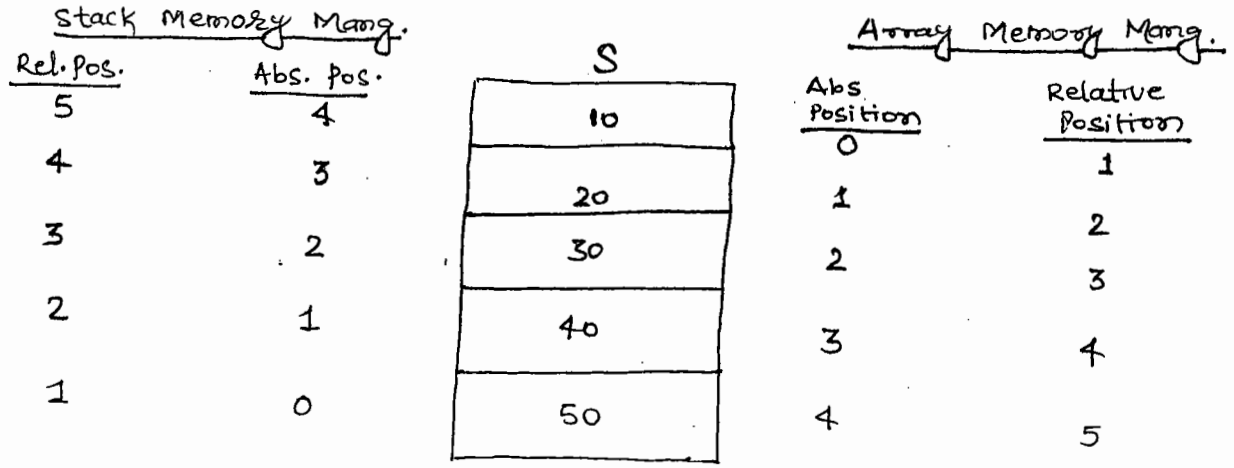
  Java.util.Stack;

- creating a stack is nothing but creating an object of stack class -

```
Stack s = new Stack();
```

- Stack is the subclass of vector so; that all the methods of vector are inherited into stack class.

- <u>memory management of stack</u>



SOP(S); // 10 20 30 40 50

L ← associativity (LIFO) R

S ⇒ object of Stack.

- <u>Profile of Java.util.Stack;</u>

Constructor

(1) - Stack()

Instance Method

(2) Public boolean empty()

(3) Public void push(object)

(4) Public object pop()

(5) Public object peek()

(6) Public int search(object)

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- constructor (1) is used for creating an object of stack.
- method (2) returns true provided stack is empty otherwise it returns false.
- method (3) is used for inserting an ele. into stack

- method (4) is used for removing an element from the stack.

- method (5) is used for extracting Top-most element from the stack.

- method (6) is used for searching an elements in the stack, if the element is found it return stack relative position of that element. (see-above-diagram) if element is not found then it returns -1.

Q. WAJP which illustrate the concept of stack & its operation ?

```
// StackDemo.java
import java.util.*;
class StackDemo
{
  psvm (String[] args)
  {
    Stack s = new Stack();
    sop("content of s =" +s); // []
    sop("size of s =" +s.size()); // 0
    sop("is a empty ? =" +s.empty()); // true
// add the element to s
      s.push(10);
      s.push(20);
      s.push(30);
      s.push(40);
      s.push(50);
    sop("content of s =" +s); // [10 20 30 40 50]
    sop("size of s =" +s.size()); // 5
    sop("is a empty ? =" +s.empty()); // False
// delete topmost element
      sop("deleted element =" + s.pop()); // 50
    sop("content of s after pop =" +s); // [10 20 30 40]
```

```
// extract the top most element
        sop("Top most element=" + s.peek());  // 40
        sop("content of s after peek =" + s);  // [10 20 30 40]
// Search 10 and 100
        int srp1 = s.search(10);
        sop("stack Position of 10 =" + srp1);  // 4
        int srp2 = s.search(100);
        sop("stack Position of 100 =" + srp2);  // -1
    }
}
```

Note :-

- At any Point of time to organize the element to LIFO Format we need to choose the Concept of stack as a collection Framework Variable.

4. Dictionary

- It is one of the 2D Legacy CFW Abstract class present in Java.util.*.
- Dictionary class object organizes the data in the form of (K, V).
- Dictionary class object displace the data in random order.
- The functionality of dictionary is more or less similar to Map interface; but dictionary class object belongs to synchronize & Map interface object is belongs to Non-Sync.

Profile of Java.util.Dictionary

IM :-

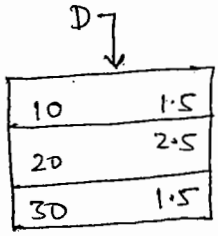
1. Public int size()
2. Public boolean isEmpty()
3. Public void Put(object, object)
4. Public object get(object)
5. Public void remove(object)

For more explanation see the Method of Map

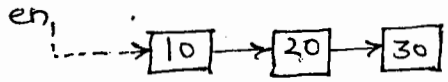
6. Public Enumeration keys()

- method ⑥ is used for extracting set of keys from any 2-D Legacy CFW variable.

ex:- sop(D); // {(-), (-), (-)}




```
Enumeration en = D.keys();
```




```
while (en.hasMoreElements())
  {
    Object kobj = en.nextElement();
    Object vobj = D.get(kobj);
    sop(vobj + " is the val. of" + kobj);
  }
```

- Dictionary class containing purely abstract method.

5. Hashtable :-

- Hashtable is one of the 2-D Legacy CFW predefined sub-class of Dictionary so that all the methods of dictionary are overridden in Hashtable.
- Hashtable class obj. organizes the data in the form of (K, V) by following hashing Mechanism.
- We can't determine in which order Hashtable class object displays the data i.e. undetermined form.
- The functionality of Hashtable is more or less similar to Hashmap but Hashtable class object belongs to sync. and never allows null values for (K, V) where as Hashmap class object belongs to Non-Sync. and allows Null values (K, V).

Profile of Java.util.Hashtable

Constructor-

① Hashtable ()
② Hashtable (int)

- Constructor ① is used for creating an object of Hashtable without specifying no. of entries to be created.
- Constructor ② is used for creating an object of Hashtable by specifying initial no. of entries to be created.

Q. WAJP is should organize the state and capitals in a CFW variable find the capital Name by entering the state?

```
//StatesCap.java
Package SC;
import java.util.*;
Public class StatesCaps
{
  Hashtable ht = new Hashtable ();
  Public void orgStatesCaps ()
  {
    ht.put ("JH", "RANCHI");
    ht.put ("BR", "PATNA");
    ht.put ("MP", "BHOPAL");
    ht.put ("MH", "MUMBAI");
    ht.put ("WB", "KOLKATA");
  }
  Public void dispStatesCaps ()
  {
    orgStatesCaps ();
    sop ("_____");
    Enumeration en = ht.keys ();
    while (en.hasMoreElements ())
    {
      object state = en.nextElement ();
      object cap = ht.get (state);
      sop (state + "  " + cap);
    }
    sop ("_____");
  }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
Public void findStateCaps()
{
 dispStateCaps();
 Scanner s = new Scanner (system.in);
 SOP("Enter the state name:");
 String state = s.nextLine();
//Look/search the caps by passing the state
   object cap = ht.get(state);
   if (cap != null)
   {
    SOP (cap+" is the capital of"+state)
   }
   else
   {
    SOP("No idea for the capital of "+state)
   }
}
```

Q. Write the main Java program for the above program?

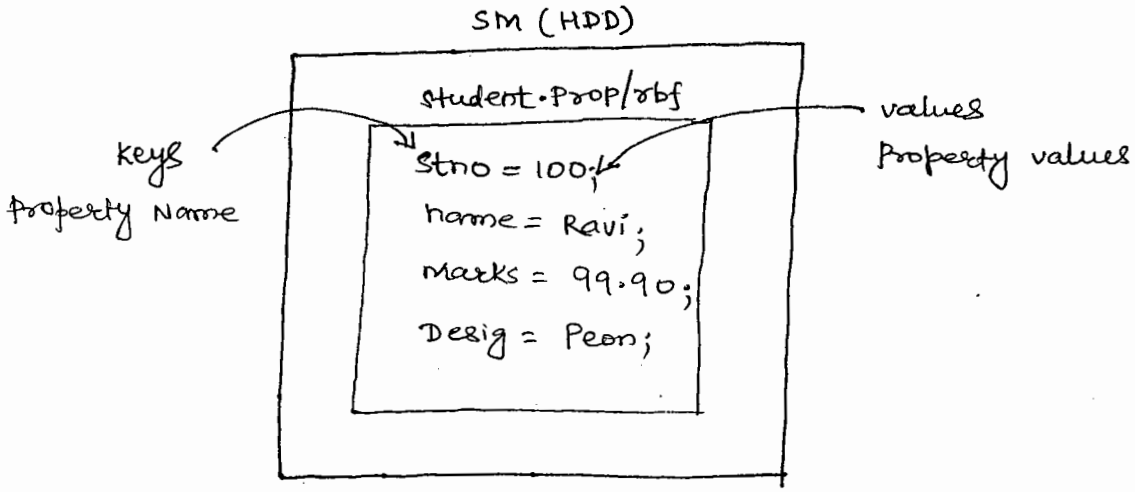
```
//StateCapsDemo.Java
import sc.StatesCaps;
class stateCapsDemo
{
 psvm (String [] args)
 {
  new StatesCaps().FindStateCaps();
 }
}
```

Limitation of Hashtable :-

- Hash Table class object unable to read the data from properties/ resource Bundle file.
- Hash Table class obj unable to read data from environmental Variable names, and Env. variable values.
- Hash Table class object unable to develop flexible application.
- To overcome these Limitation of Hash table we use another predefine class called java.util.properties

☆ Properties / Resource Bundle File :-

- A Properties file is one of the Text file to be created in any Text Editor and it should be saved as any file name with an extension either .prop or .rbf.
- A properties file always organize the data in the form of (key, value).



- Properties Files always resides in Secondary Memory.
- To Read the data from property file first it must be opened in read mode with the help of FileInputStream class.
- FileInputStream fis = new FileInputStream("Student Prop");

Properties class :-

- It is one of the Predefine 2D legacy CFW class of HashTable and Present in Java.util.*;

Adv. of properties class

1. - Properties class object makes us to read the data from Properties file/ Resource Bundle file.
2. - Properties class object makes us to read environmental variable name & Environmental variable values.
3. - Properties class object makes us to develop flexible application.

Creating Properties is nothing but Creating an object of Properties class.

```
Properties p = new properties();
```

p ← resides in Main memory

K	V

✱ Profile of Java.util.properties Class

Constructor

① properties()

Instance Method

② public void load (FileInputStream)

③ public String getProperty (String)

- Constructor ① is used for creating an object of Properties class.

```
properties p = new Properties();
```

- method ② is used for loading & Transferring the content of properties file into the object of properties class by opening the properties file in read mode.

```
FileInputStream fis = new FileInputStream("student.Prop");
p.Load(fis);
```

p ← MM

Stno = 100
name = Ravi
marks = 99.9

- Method ③ is used for reading properties values (value of values) by passing Property name (keys).

eg- String sno = P.getProperty ("stno");
String Sname = P.getProperty ("name");
String marks = P.getProperty ("marks");

Q. write the Java Program which will read the data from property file by using Property class ?

```
// Prop.Java
// Before compiling executing this prog. we create the prop.file
import Java.util.*;
import Java.io.*;
class Prop1
{
 psvm (String [] args)
 {
   try
   {
     Scanner s = new Scanner(System.in);
     sop ("enter the prop file name");
     String fname = s.nextLine();
     FileInputStream fis = new FileInputStream (fname);
     properties p= new properties ();
     p.load (fis);
     sop ("Student Number =" + P.getProperty ("stno"));
     sop ("Student Name =" + P.getProperty ("name"));
     sop ("Student Marks =" + P.getProperty ("marks"));
   } catch (FileNotFoundException fe)
     {
       sep ("prop file does not exists");
     }
     catch (Exception e)
     {
       sep (e)
     }
 }  }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. write a Java Prog. which will read Env. variable names & Env. variable values.

```
// Prop2.Java
import Java.uit.*;
class Prop2
{
  PSVM (String [] args)
  {
    Properties P = System.getProperties();
    Enumeration en = P.keys();
    while (en.hasMoreElements())
    {
      object evn = en.nextElement();
      object evv = P.get(evn);
      SOP (evn+" -> "+evv);
    }
  }
}
```

The end

CFW

Multi Threading

- The purpose of multithreading is to provide concurrent execution simultaneously or parallel execution of multiple operation.
- Flow of control is known as "thread"
- A flow of control is purpose has to executing the logic of Java Program which is written in the form of user-define method concurrently.
- If any Java program contains multiple thread then it is called multiple thread program.
- The Language like c, c++, pascal, COBOL etc are treated as single threaded Modelling Language because there execution environment provides single flow of control, sequential execution and takes more execution time and they donot contain any library for development of multithreaded appl^n.
- The language like Java, .Net are treated as multithreaded Lang. because there execution environment provides component execution; multiple flow of control, less execution time & contains an effective API for development of multithreading application.
- In Java programming to develop multithreading application we use the following API :-

  a.) Java.lang.Thread (class)

  b.) Java.lang.Runable
- whenever we write any Java program there exist two types of thread ; they are -

  1. foreground Thread
  2. Background Thread.
- A foreground thread always execute logic of Java program. which is written in the form of unsatisfied method.
- A Background thread is always monitor the execution status of foreground thread.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- By default there exist single foreground thread & single background thread. whereas programmatically we create multiple foreground thread & recommended to create single background thread.
- The realtime implementation of multithreading is that to develops real world server software. In otherwords most of the real world server software (tomcat, weblogic, etc) are developed by 3rd party vendors in Java Language with Multithreading Concept.
- Here multithreading concept in Java is one of the specialized form of multitasking concept of OS.

Q. How do you Justify each & every Java Program is multithreaded ?

- Whenever we execute any Java Program, Logic of the Java program is executed by one of thread known as <u>Foreground thread</u>.

  To Monitor the execution of foreground thread one more thread created internally known as <u>background thread</u>.

  So by default there exist two thread in Java program & hence each & every Java Program is multithreaded.

Q. What are the difference between Program & Process ?

Program	Process
1.) A program is a set of optimized instruction.	1.) A program is under execution is called Process.
2.) Program resides in secondary memory for long time.	2.) A process resides in Main Memory for limited span of time.
3.) Programmatically a program can be Considered as a static object.	3.) A process can be Considered as dynamic Object.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Average Bakery
Opp CMC Bakery Jail Road,
Anantapur (A.P.)

Process Based VS Thread Based Application :-

(OR) Non-multitasking vs Multitasking Application

PBA	TBA
1. In PBA there exists single flow of control.	1. In thread based Application there exists multiple flow of control.
2. All the single threaded Modelling Lang. related Appl^n are treated as PBA. i.e. C, C++, etc.	2. All multi-threaded Modelling Lang. related Appl^n are treated as TBA. ex:- Java, .Net etc
3. Context switch is more in PBA.	3. Context switch is less in TBA.
4. For each & every sub program (function) there exist a seperate address space.	4. Irrespective of no. of sub-prog, there exists a single address space.
5. All PBA are treated as heavy weight components.	5. All TBA are treated as light weight components.
6. PBA takes more Execution Time. (∵ Context switch is more)	6. TBA takes less execution time. (Because context switch is less)
7. PBA provides only sequential execution & never provides Concurrent execution.	7. TBA provides both sequential as well as Concurrent execution.

- **Defination of Address Space** -

It is an amount of temporary space created by O.S. on stack memory for the execution of the method.

- **Defination of Context Switch** -

The mechanism of switching control of the CPU from one address space to another address space; it called Context Switch.

for the Best application development Context Switch must be less.

Heavy weight components takes more execution & Light weight component takes less execution time.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

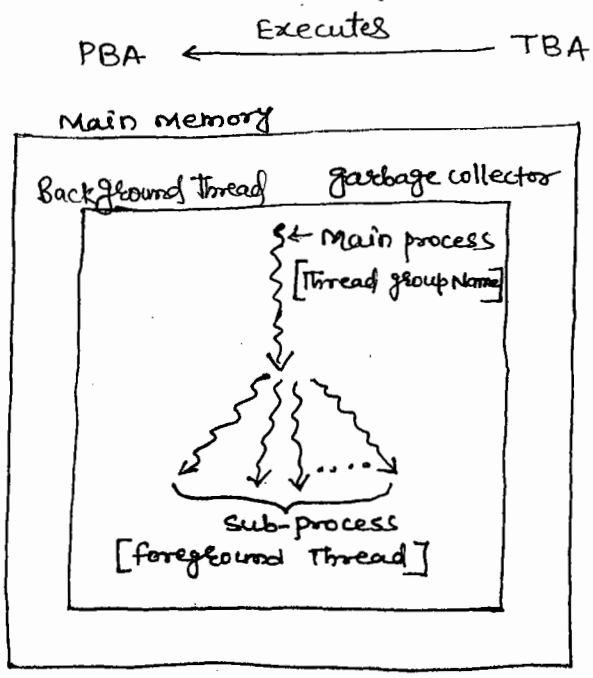
*

$$\boxed{\text{Execution Time} = \text{I/O Burst Time} + \text{C.P.U. Burst Time}}$$

- An amount of time taken by the program from I/O devices for completing I/O operations is called <u>I/O Burst time</u>.
- An amount of time required by the program from CPU for completing processing operation is called <u>CPU Burst Time</u>.

<u>Sharp Note-1</u>:-

- E & E TBA is one of the PBA and in which one main process will be created and the main process ~~interms~~ creates sub-process(es). In Java programming point of view the main process is called thread group name and sub-processes are called foreground threads.
- Consider the following diagram -

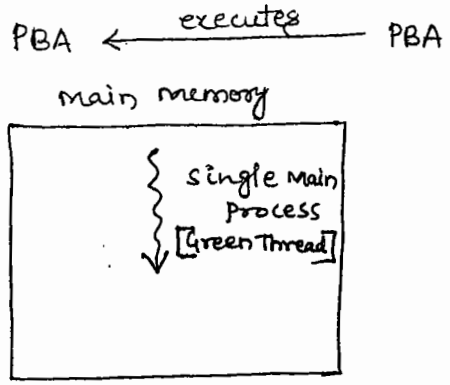


SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

<u>Sharp Note-II</u>:-

- ~~A PBA can be treated as PBA but not treated as~~
- When we execute the PBA then it will be considerd as PBA only but not TBA, because there exist only one main process (single flow of control - green thread).

SRI RAGHAVENDRA XEROX
Software
Beside



- **States (or) Life cycle of a Thread :-**

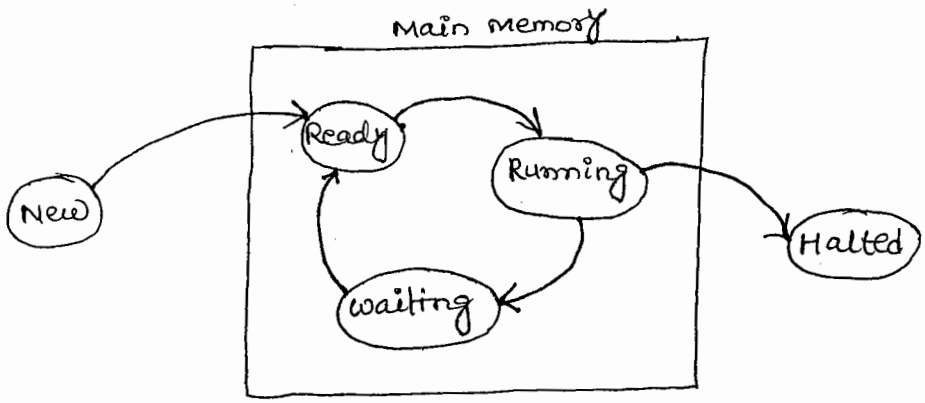
The different stages or Breaking Points occuring in the execution of logic of Java Program which is written in the form of Java program; is called State of a thread.

(OR) - The diff. breaking point in the execution of logic of Java prog. by the thread are called lifecycle or State of a Thread.

- Thread states are classified into fives types —
  1. New state
  2. Ready state
  3. Running state
  4. Waiting state
  5. Halted state

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Thread state chart diagram —



1. New State :-

In this state thread is created and it is about to enter into Main memory.

2. Ready state -

In this states thread is entered into main memory. memory space is allocated for the thread. Thread is waiting first time for cpu.

3. Running State -

In this State thread is under the control of cpu.

5- Halted State -

In this state the thread completed its entire execution.

4. Waiting State -

A thread is said to be in waiting state if and only if has to satisfy any one of the following factors :-

1. Thread is coming to waiting state for the remaining CPU Burst time.
2. Suspending the currently executing thread.
3. Making the currently executing thread to sleep for a period of time interms of ms. (1 sec = 1000 ms).
4. Making the currently executing thread to wait with time.
5. Making the currently executing thread to wait without time.
6. Joining the thread.

Hence, The new state & Halted state are called out-Memory state and whose execution status is false.

Whereas ready, running & waiting state are called in-memory states and whose execution status is True.

In The Java programming environment all the threads are executing Concurrently with the difference of ms of time by following Round-Robin algorithm.

* **Creating a thread** — To create a Thread in Java programming; we have two approaches. They are —
  - By using Java.lang.thread class.
  - By using Java.lang.runnable Interface.

- **By using Java.lang.thread class** —

By using Java.lang.thread class we can create its object in the three way. They are :-

(a) By using New operator :-

```
Thread t1 = new Thread();
```

(b) By using Factory method :-

```
Thread t2 = Thread.currentThread();
```

(c) An object of sub-class of Java.lang.Thread is an object of Java.lang.Thread.

```
class Th1 extends Thread
{
  ==
  ==
}
Th1 t1 = new Th1();
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Here t_1 is an object of Th_1 and Th_1 is sub-class of Jav.lang.thread class. Indirectly t_1 is an object of Java.lang.Thread class.

Profile of Java.lang.Thread class.

Data Members

(a) public static final int MAX_PRIORITY (=10)

(b) public static final int NORM_PRIORITY (=5)

(c) Public static final int MIN_PRIORITY (=1)

- The above data members are called Thread Priority Modifiers. The purpose of Thread Priority modifiers is that makes the JVM to understand which thread to execute first, second & last.
- The default Priority of the thread is NORM_PRIORITY (_)

Constructors

① Thread () :

This constructor is used for creating an object of thread class without specifying user-friendly name to the thread.

ex:- Thread t_1 = new Thread ();

object Name/refrence var.

- Here default name of Thread is "Thread_0".
- In general if we create n no. of object with this default Constructor whose default name starts with Thread_0, Thread_1, Thread_2 ..... Thread_n-1.
- giving default name in the thread object is not recommended and recommended to give user-friendly Name to the thread.

② Thread (String) :-

It is used for creating an object of Thread class by giving userfriendly Name to the Thread.

ex:-

Thread t_1 = new Thread ("satyya");

object Name/variable Name. — user friendly name to the thread

③ Thread (Runnable) - "object Parametrized constructor."

This constructor is used for converting Runnable interface object into Thread class object for making use of all the method of Thread class (specially start method) to enter into the overridden run() of subclass of Runnable interface.

ex:- Th$_1$ t_1 = new Th$_1$();

(or)

Runnable t_1 = new Th$_1$();

t_1.start(); // Invalid

ref.

Thread t_{11} = new Thread (t_1);

t_{11}.start();

④ Thread (Runnable, String)

The functionality of this constructor is adjactly similar to above Constructor. But additionally with this Constructor we are giving User Friendly Name to the thread.

ex:- Runnable t_1 = new Th_1 ();

// t_1.setName ("Sathya"); invalid

Thread t_{11} = new Thread (t_1, "Sathya");

String tname = t_{11}.getName();

User Friendly Name

sop (tname); // Sathya

- Instance Methods

① Public final void setName (String)

② Public final String getName ()

- The above method are used for setting the UserFriendly Name of thread & getting the Name of the thread.

ex:- Thread t_1 = new Thread ();

→ Thread-0

String tname = t_1.getName();

sop (tname); // Thread-0

t_1.setName ("PATNA");

tname = t_1.getName ();

sop (tname); // PATNA

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- Mutator Method are those which are used for changing the value of the object and they start with setXxxx (-).
- Inspector Method are those which are obtaining the values from objects and they starts with getXxxx ().

③ Public final void setPriority (int);

④ Public final int getPriority ();

The above methods are used for setting the Priority to the thread and getting the Priority of the thread.

ex:- Thread t_1 = new Thread ("Sathya");

int pv = t_1 getPriority ();

sop (pv); // 5 ⇒ N.P.

```
t1.setPriority (Thread.MAX_PRIORITY);
PV = t1.getPriority ();
SOP (PV); // 10 MAX_PRIORITY
```

(5) Public final boolean isAlive ()

- This method is used for checking the executing status of the thread. This method returns True provided the thread is in Ready, Running, waiting states. This method returns false provided the thread is in New, Halted states.

** (6) Public void run ()

- This method is used for defining the logic of the thread, In other words this method is defined with null body and as a Java Programmer we override the null body run () with the Logic of Java Program.

*** (7) Public final void Start ().

- When we call Start () upon the thread class object then Start () will perform the following internal operation:-

1.) It Moves Thread from New State to Ready State.

2.) It provides internal services of Multithreading. (Concurrency, synchronization, inter thread comm.).

3) It automatically calls run () ; which contains the logic of the thread.

- To develop any multithreading appl^n we nee to choose one predefine class it must extends Java.lang.Thread and need to Override null body method of Thread class.

```
ex:- class Th1 extends Thread
     {
      Public void run () // overridden
      {
        ===  // Logic of thread.
      }
     }
```

ex:-

```
Th1 t1 = new Th1 ();  // New state
boolean b = t1.isAlive ();
    sop(b);  // false
    t1.start();  // Ready state
    b = t1.isAlive ();
    sop(b);  // true
```

Note -

It is not recommended to call run() directly and always recommended to call through start() ; because start() provides internal services of multithreading.

8.) Public final void suspend ()

- This method is used for suspending the currently executing thread.
- When the thread is suspend its execution, whose temporary exe. detail will be store in PCB and it will be entered into waiting state.

```
t1.suspend ()
↑ object of sub class of thread class.
```

9.) Public final void resume()

When we use resume() upon the thread class object the suspended thread will be transfer from waiting state to ready state and starts its execution where it left off before its suspention by retreiving temporary execution detail from PCB.

```
t1.resume ()
```

Note - PCB (Process Control Block) :- It is a Buffer. created by operating system. when the thread is entered first time in ready state and it is used for storing temporary execution detail of a thread.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

10.) Public final void stop()

This method is used for stopping or terminating the currently executing thread. The stopped threads entered into Halted state. and when there start execution it starts from the begining.

11.) Public final void Join() throws Java.lang.InterruptedException

- This method is used for Joining the completed Foreground Thread as a single unit. To improve the performance of multithreading application.

- The default env. of multithreading says as when Foreground Thread are completed there execution, they are individually collected by threadgroupName and handover to garbage collector individually it is one of the time consuming process. To overcome this time consuming process we use Join method. To Join all the completed threads in a single unit (in waiting state).
  and all at once collected by Thread groupName and handover to garbage collector.

- This method throws a predefined Exception called Java.lang. Interrupted Exception.

- When the Interrupted Exception occurs it is one of the checked Exception.
  It never across in standalone machine and rarely occurs in busy env. like client-server env due to mis-memory mngt of server side O.S.

  Situation:-

  Let us assume there exist n no's. of thread Created, started there execution, completed there execution, and (n-1) threads are join and still n^{th} thread is under execution.
  At this points of time due to mis-memory management of server-side O.S thread group Name is attempting to Collect all the thread including n^{th} thread,

w.r.t. t_n JVM generates a predefine exception called java.lang.InterruptedException.

ex:-

```
try
{
    t1.join();
    t2.join();
    :
    tn.join();
} catch (InterruptedException ie)
{
    sep("problem in Thread Execution");
}
```

12. <u>Public ThreadGroup getThreadGroup()</u>

This method is used for obtaining the thread group information (ThreadGroup Name, priority) of foreground Thread.

ex:-

```
ThreadGroup tg = t1.getThreadGroup();
        SOP(tg); // ThreadGroup [name = main, max Pn = 10]
ThreadGroup tgp = tg.getParent();
        SOP(tgP); // [system]
ThreadGroup tgpp = tgp.getParent();
        SOP(tgpp); // null
```

13. <u>Public Thread.State getState()</u>

This method is used for obtaining the name of the thread state.

ex:-

```
Thread t1 = new Thread();
Thread.State ts = t1.getState();
        SOP(ts); //New
boolean b = t1.isAlive();
        SOP(b); // false
```

Static Method -

(1) Public static final Thread CurrentThread ()

This is one of the static factory method and it is used for obtaining the thread which are default running in Java Environment.

ex:-

```
Thread t1 = Thread.currentThread();
    sop(t); // Thread(main, 5, Main)
t.setName("Sathya");
    sop(t); // Thread [Sathya, 5, Main]
```

FGT Priority Thread GroupName

(2.) Public Static Final void Sleep (Long) throws Java.lang.InterruptedException

- This method is used for making to currently executing thread to sleep in terms of ms. once we call sleep() the thread is entered into waiting state by placing the temporary details in PCB, and it comes to ready state when sleep time is completed.
- As long as the thread is sleeping further operations, statement will be not performed/executed.
- This method throws a predefine exception called Java.lang.InterruptedException
- When InterruptedException occurs

It is one of the checkedException and never across in Standalone Mechine and rarely occurs in busy environment like client server environment due to mis-memory management of server side O.S.

Situation -

Let us assume first Program thread are sleeping in some location of serverside O.S. due to mismemory management of server side O.S. The second program thread are attempting to sleep in the same location where the First Program Thread are sleeping at the point of time w.r.t. first program thread JVM generates a Predefine Exception called Java.lang.InterruptedException.

Ex:
```
try
{
    ========
    ========
    Thread.sleep(1000);
    ========
}catch (InterruptedException ie)
{
    sep ("problem in thread Exception");
}
```

Q. Write a Java program which will display plimilary information about the Thread such as Threads which are by default running, whose execution status, state name of those thread, Thread Group name of those thread, default name of programmer created foreground thread, whose execution status, whose status name, Thread priority modifier values etc.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
// Thdemo1.java
class Thdemo1
{
 psvm (String [ ] args)
 {
  Thread t = Thread.CurrentThread();
   sop("default thread =" + t); // Thread [main, 5, main]
    t.setName ("sathya");
    sop ("default threads after modification =" + t); // Thread
                                                        [sathya, 5, main]
    sop ("exe. status of t =" + t.isAlive()); // true
    Thread.State ts = t.getState();
     sop("statename of t (default FGT) =" + ts); // RUNNABLE
// obtain the thread group info of default FGT
        ThreadGroup tg = t.getThreadGroup();
         sop(" thread group of tg" =" + tg); // ThreadGroup
                                                [name = main  max Pri = 10]
// obtain the Parent of ThreadGroup - tg
        ThreadGroup tgp = tg.getParent();
           sop ("Parent of thread group tg =" + tgp);
           // ThreadGroup [name = System  MAX Pri = 10]
```

```
// info about Prog created FGT
    Thread t1 = new Thread();
    SOP ("default name of prog created FGT =" + t1.getName());
                                                    // Thread-0
    SOP ("exe status of t1 =" + t1.isAlive); // False
    SOP (" state name of t1 = + t1.getState ()); // NEW
// Thread Priority modifier value
    SOP (" val of max pri =" + Thread.MAX_PRIORITY);
    SOP (" val of NORM Pri =" + Thread.NORM_PRIORITY);
    SOP (" val of MIN Pri =" + Thread.MIN_PRIORITY);
  }
}
```

Internal flow of Thread :-

When we write any multithreading appl^n there exist many no. of threads. It is mandatory to java programmer to find internal flow of the threads.

The following sequence of steps gives internal flow of thread:-

1.) Java starts Execution.

2.) JVM creates Thread Group Name (TGN) & it always resides & executed stat. Present in main().

3.) TGN creates foreground Threads (FGTs) always meant for execution logic of java program which is written in the form of run().

4.) TGN dispatches FGT (s) to their respective run().

5) FGT(s) executes respective run() and gives result to TGN — either all at once or one by one.

6) TGN resives the result from FGT(s) either all at once or, one by one.

7) TGN gives of java program to appl^n user (Programmer)

8) TGN collects FGT(s) and handover to gc. (garbage collector)

9.) JVM collects TGN and hand over to GC.

10) Java stop execution.

Q. write a Java Program which will print one to 10 no's after each and every seconds.

```
// Thdemo4.Java
class Th1 extends Thread
{
 public void run()
 {
  try
  {
   For (int i=1 ; i<=10; i++)
   {
    sop ("val of i="+i);
    Thread.sleep (1000);
   } // For
  } catch (InterrupedException ie)
   {
    sep ("prob. in thread exe....");
   } //run()
 } // Th1 - BLC
Class Thdemo4
{
 Public static void main (String [ ] args)
 {
  Th1 t1 = new Th1();
  SOP ("exec status of t1 before start =" + t1.isAlive());
      t1.start();
  SOP ("exec status of t1 after start =" + t1.isAlive());
//  try
  {
   Thread.sleep (5000);
  }catch (InterruptedException ie)
   {
    sep ("prob in thread exec....");
   }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
SOP ("exec status of t1 during execution =" + t1.isAlive());
try
  {
    Thread.sleep (5001);
  }catch (InterruptedException ie)
    {
      sep ("prob in thread Exce...");
    }
    SOP ("exe status of t1 after completion =" + t1.isAlive());
//  SOP("state name =" + t1.getState()); // TERMINATED
  }
}
```

Solution ①

↑ solution 1.

NOTE

The following solution gives Printing the execution status of FGT False after its completion irrespective of the time it sleep.

```
While (t1.isAlive())
{
}
SOP ("exe status of t1 after infinite loop false (completion) ="
                                        + t1.isAlive());
```

Solution ②

• we write the above statement in main().

NOTE :-

The following solution also makes to print the execution status of the Foreground Thread has false after its completion (after Join)

```
try
{
  SOP ("TGN is waiting for FGT to Join");
        t1.Join();
}catch (InterruptedException ie)
  {
    sep ("prob in thread exec....");
  }
```

solution ③

```
SOP ("exec status of t1 after join (completion) =" + t1.isAlive());
```

- The above code segment must be written within the main()

By using Java.lang.RunnableInterface -

It is one of the functional interface which contains only one Abstract method whose prototype is given below -

```
public abstract void run()
```

- Thread class contains null body run() and RunnableInterface contain abstractRun().
- Industry always recommended to override abstract run() of RunnableInterface.

Let us consider the following invalid code segment :-

```
class Th1 extends Sathya, Thread // Invalid bcoz one derived
{                                class can extends only one
                                 Base class.
}
```

- To avoid the above multiple inheritence problem we must make use of Runnable interface and the above invalid code segment rewritten as follows :-

```
class Th1 extends Sathya implements Runnable
{
    public void run() // overridden Method
    {
        ___
        ___   // Logic of thread
        ___
    }
}
Th1 t1 = new Th1();
  // t1.Start(); invalid
Thread t11 = new Thread(t1);
        t11.Start();
```

- Runnable interface obtain only one abstract method and whose signature is given below –

  public abstract void run()

  hence, To develop any multithreaded application choose a user define class, it must implements Java.lang.Runnable interface and needs to override run().

☆ Note :–

In the Java API thread class implements runnable interface and provides null body run().

Industry is not recommended like that, highly recommended to override abstract run() of runnable interface but not null body run() of thread class.

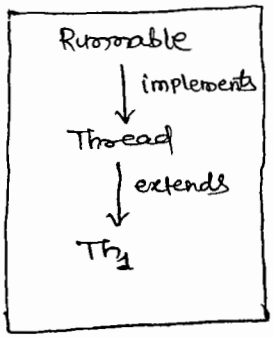
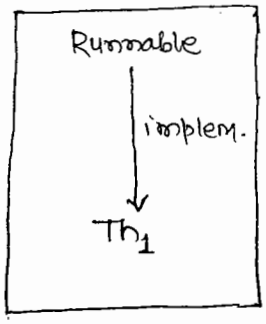


fig-(i)

[Not recommended]



fig[ii]

[Recommended by Industry]

Q. Letus consider the following code segment :-
(which will print FGT Name in the context of run() and it display any textual Msg character by character after each & every second.)

```
// ThDemo6.Java
import java.util.Scanner;
class Th1 implements Runnable
{
  Th1()
  {
    SOP ("Th1 - DC");
    Thread t = new Thread( this, "Sathya"); //converting current
                                              Class obj (Runnable intf. obj) into
                                              Thread class obj.
```

Q. Develop a thread Based application which will compute sum & substraction two' no's concurrently ?

```
//Thdemo6.java
Import Java.util.Scanner;
class sum extends Thread
{
 int a, b, c;
 void set (int a, int b)
  {
   this.a = a;
   this.b = b;
  }
  Public void run()
  {
    c = a+b;
    SOP ("sum (" +a+ "," +b+") =" +c);
  }
}
  class sub implements Runnable
  {
    int a, b, c;
    void set (int a, int b)
     {
      this.a = a;
      this.b = a;
     }
    Public void run ()
    {
      c = a+b;
      SOP ("sub (" +a+ "," +b+")" +c);
    }
  }
      class Thdemo6
    {
      PSVM (String [] args)
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
{
// accept values from keyboard
    Scanner s = new Scanner (System.in);
    sop (" Enter the value for first");
    int x1 = Integer.parseInt (s.nextline ());
    sop ("Enter the value for Second");
    int x2 = Integer.parseInt (s.nextline ());
// create the thread
        sum s1 = new sum ();
        sub S2 = new sub ();
// set the values to the thread
        S1.set (x1, x2);
        S2.set (x1, x2);
// dispatch FGTs (S1, S2) to compute sum & sub operations
            S1.start ();
            // S2.start ();  invalid
            Thread S22 = new Thread (S2);
                S22.start ();
// Join FGT into single unit
        try
        {
            S1.Join ();
            S22.Join ();
        } catch (InterruptedException ie)
            {
            sep (" problem in thread execution");
            }
        }
```

NOTE:-

When we want to execute multiple independent operations concurrently, then we must choose multiple SubClasses of thread Runnable Interfaces and we must override multiple run() in multiple subclass with multiple independent op. logic.

Create the following code segment, and in which we are printing name of FGTs in the context of run().

```
//Thdemo7.java
class Th1 implements Runnable
{
 Thread t1;
 Th1()
 {
  sop("i am Th1()");
  t = new Thread(this, "sathya");
  t.start();
 }
 public void run()
 {
  sop("i am from run()");
//String tname = t.getName(); //-sol1 to print name of FGTs
 Thread ct = Thread.currentThread(); //-sol2 print name of FGT
 sop("FGT name in run()=" + ct.getName()); // sathya
 }
}
class Thdemo7
{
 psvm (String[] args)
 {
  sop("I am from main()");
      new Th();
 }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. WAJP which will implements Banner Animation (or) write a thread based appl^n for Scrolling of a given Message.

```
//ani.java
import java.awt.*;
import java.applet.*;
/* <applet code="ani" height=500 width=500> </applet> */
public class ani extends Applet implementation Runnable
{
  String msg = "RED IN COLOR";
  public void init()
  {
    setBackground (color.red);
    setForeground (color.Blue);
  }
  public void start ()
  {
  Thread t = new Thread (this);
// this is refers to current class object. which implements
"Runnable" interface and convert into Thread class obj.
      t.start();
  }
  public void Paint (Graphics g)
  {
  Font f = new Font ("arial", Font.BOLD, 60);
    g.setFont (f);
    g.drawString (msg, 100, 100);
  }
  public void run()
  {
    try
    {
      while (true)
```

```
}
 char ch = msg.charAt(0);
 msg = msg.substring(1, msg.length());
 msg = msg + ch;
 Thread.sleep(500);
 repaint();
}

} catch (InterruptedException ie)
    {
     sep("problem in thread");
    }
} //run()
} //ani-class
```

Run the above program

```
Appletviewer Ani.java
```

Compile

```
Javac Ani.java
```

Q. write a thread based Application which will generate automatic incrementation of numbers.

```
//AutoInc.Java
import Java.awt.*;
import Java.applet.*;
/*<applet code="AutoInc" height=500 width=500> </applet> */
Public class AutoInc extends Applet implements Runnable
  {
   int i=0;
   String msg = "WELCOME TO HYDERABAD";
   public void init()
   {
    SetBackground (color.yellow);
    SetForeground (color.Blue);
   }
   public void start()
    {
     Thread t = new Thread (this);
       t.start();
    }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
Public void Paint (Graphics g)
{
 Font f = new Font ("arial", Font.BOLD, 80);
  g.setFont (f);
  g.drawString (msg, 100, 100);
}
 Public void run()
 {
   try
    {
     while (true)
      {
        i++;
        char ch = msg.charAt (0);
        msg = msg.substring (1, msg.length());
        msg = msg + ch;
        Thread.sleep (1000);
         repaint ();
      }
    } catch (InterruptedException ie)
     {
       system.err.println ("problems in thread");
     }
   }
 }
```

Thread Synchronization

The aim or purpose of synchronization is to build consistent application when multiple threads executes, multiple dependent operations or sharable resource related application.

In other words when multiple threads operates on same resource concurrently by default the threads will generate inconsistent result or deadlock result.

To overcome this inconsistent/Deadlock result we must apply the in-built concept of sync.

Synchronization is one of the concept of operating system and it is available in Java prog. as built-in concept to build concept appl.

Def. of sync.

- The process of allowing only one thread among multiple threads into the sharable area/resource to perform read and write operation for getting consistent result by eliminating Deadlock result, is known as synchronization.

Need of synchronization:-

Let us assume there exist a sharable variable Balance; whose initial value is "0". Let us assume there exist two threads t_1 and t_2 decided to deposite in the Balance of account. both are started there execution, and completed there execution.

The final value of Balance is either 10, 20 but not 30; which is one of the inconsistent result.

To eliminate this inconsistent result on the above problem we must apply the concept synchronization.

Synchronization applied (on above program)

Both the threads t_1 & t_2 (10,20) started there execution to deposite rupees 10,20 in Balance of account (t_1 starts first, and later t_2 start with the difference of milliseconds of time)

- JVM gives value of balance (0) to t_1 and immediatly balance variable will be locked by making thread t_2 to wait until thread t_1 complete.
- thread t_1 completed its execution, JVM unlock the Balance variable, JVM gives value of Balance (10) to thread t_2 and JVM locked Balance variable.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

- Thread t_2 also completed its execution and JVM will unlock Balance variable.
- The final value of Balance variable is 30; which is one of the Consistent result.

Hence, In the synchronization process JVM continues the process of locking and unlocking until all the threads completed there execution.

Thread Synchronization Technique

we have two types of thread Synchronization Techniques. They are –

1. Synchronized methods
2. Synchronized Blocks

1. Synchronized methods

we have two types of synchronized methods :–

ⓐ Synchronized Instance method

ⓑ synchronized static method

ⓐ Synchronized Instance method

If any ordinary Instance method accessed by multiple thread then we get inconsistent result.

To get consistent result, programmatically the ordinary instance method definition must be made as synchronized by using "Synchronized" keyword.

Syntax For Synchronized Instance Method :–

```
Synchronized ReturnType methodName (List of formal Parameters if any).
{
    Block of stat(s) – Business Logic – Thread Safety result
}
```

As long as thread is executing synchronized Instance method, JVM will lock an object of corresponding class.

ex:-

```
class Account
{
  int bal = 0;
Synchronized void deposit (int amt)
    {
      bal = bal + amt;
      sop ("current bal =" + bal);
    }
}
```

As long as the thread is executing synchronized instance deposit method (above program) JVM will locked an object of Account class.

(b) Synchronized Static Method –

If an ordinary static method accessed multiple thread concurrently then we get inconsistent result.
To eliminate inconsistent result the definition of ordinary static method must be as synchronized by using "synchronized" keyword.

Syntax:–

```
Synchronized Static ReturnType ( List of Formal Parameter if any)
 {
   Block of Stat(s) – Business logic – thread safety result
 }
```

As long as the thread is executing synchronized static method, JVM locks corresponding class.

ex:-

```
class Account
 {
   int bal = 0;
Synchronized  Static void (int amt)
   {
     bal = bal + amt;
     sop ("current bal =" + bal);
   }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

As long as the thread is executing the above synchronized static deposit method, then JVM will lock Account class.

Hence,
synchronized methods always makes the JVM ~~to~~ will lock either on object (OR) class depends on types of synchronized methods execution.

Synchronized Block :-

- Synchronized Block is an alternative Mechanism for getting consistent result instead of using synchronized Method.
- Synchronized Blocks to be written always within the context of Non-synchronized instance Method.
- Synchronized Blocks always makes to JVM locks object of current class.

Syntax :-

```
synchronized (object of current class)
{
    Block of stats - Business logic
                   - Thread safety result
}
```

Importance of Synchronized Blocks

If we inherit Non-synchronized instance method into the Context of derived class either from Base class or Base Interface and if the inherited Non-synchronized Instance method accessed by Multiple thread then we get inconsistence result.
To avoid this inconsistence result, As a derived class programmer be may attempt to write "synchronized" keyword before inherited non-synchronized instance Method because a derived class programmer can't change the prototype of Base class / Base interface method.
Hence with sync method we are unable to eliminate In-consistence result and to eliminate this inconsistence result we use an another sync technique called Sync. Block.

```
interface Account
{
 void deposit (int); // Public abstract non-syn IM
}
class SavingsAccount implements Account
{
  int bal=0;                    Inh. non.sync. IM
Public void deposit (int amt)
{
 Synchronized (this)
 {
  bal = bal + amt;                  sync. Block
   SOP ("curr. bal =" +bal);
 }
}
}
```

Hence, Thread sync. Techniques always provide thread safety result.

Q. WAJP which illustrate the Concept of Synchronization.

Prob Stmt :- Let us assume there exist an account with the Balance and whose initial value is "0".

- An account we can perform operation like depositing & Bal enquiry
- Let us assume there exist n no's of customer who decided to deposit rupees 10 by each in the Bal of account concurrently. Obtain the Consistance result during the concurrent execution of multiple customers. depositing an account in the Bal of account.

```
// sync.java
class Account
{
 Private int bal=0;
Synchronized void deposit (int amt)
 {
   bal = bal + amt;
   SOP("curr. bal =" + bal);
 }
int getBal()
 {
   return bal;
}  }
```

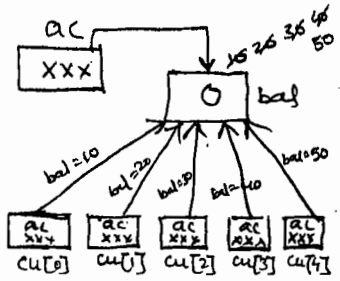
SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
class cust extends Thread
 {
  Account ac;
  Cust(Account ac)
   {
    this.ac = ac;
   }
  public void run()
   {
    ac.deposit(10);
   }
 } //cust
 class sync
  {
   public static void main(String [] args)
   {
 //create a single object of Account class
          Account ac = new Account();
 // create Array of cust class object i.e. threads
          cust cu[] = new cust[5];
 // give single ref. of Account class to array of cust class objects
        For(i=0; i<cu.length; i++)
        {
         cu[i] = new cust [ac];
        }
 // dispatch the array of cust threads to deposit Rs:10/- by each
        For(i=0; i<cu.length; i++)
          {
           cu[i].start();
          }
 // Joining all threads in a single unit
       try
        {
         for(int i=0; i<cu.length; i++)
          {
           cu[i].join();
          } //For
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
    } catch (InterruptedException ie)
      {
        sep (" problem in thread Execution");
      }
  // obtain final bal
        sop (" Total Balance =" + ac.getBal());
      }
  }
```

Synchronization view of above prog is -



NOTE-1 :-

when we execute any synchronization based applications (sync.java) on Mono OSs or single user OSs (DOS, Win 3.1, win 95) then it is mandatory to Java programmer to write "Synchronized" keyword (Java based synchronization) otherwise we get inconsistant result because even through synchronization concept present in OS which is not implemented in Mono OSs.

NOTE-II -

When we execute synchronization based Appl^n (Sync.Java) on multiuser/multithreaded OSs (windows 98, win XP, ---Unix, ---etc) then it is optional to write "synchronized" keyword because synchronization concept of OS is implemented in multithreaded OSs. Hence multithreaded OSs contains in-built synchronization and it provides final result as consistant & order of updation is in-consistant result which is not a recommended process.

NOTE-III

Irrespective of type of O.S. being used by client organization, industry is highly recommended to write "synchronized" keyword (Java based synchronization) for getting final result as consistant and order of updation is consistant. which is one of the recommended process.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Inter Thread Communication
(ITC)

- ITC concept of Java is one of the specialized form of IPC concept of OS.
- ITC based applications are fastest application compared to IPC application because Thread to Thread communication is treated as light weight component and process

- Real Time implementation of ITC :-

- To develop real world universal protocol (HTTP, FTP, SMTP, POP..).
- To develop real world server software (Tomcat, websphere, weblogic).

Def. of ITC :-

The process of exchanging the information/data between multiple threads is known as ITC (or) If the output of first thread is given an input to 2nd Thread, the output of 2nd Thread is given an input to another Thread etc the consistence comm. between 1, 2, 3 Thread is known as inter thread communication.

To develop ITC based Appl^n we have the predefine methods present in Java.lang.object and these methods are called as ITC method.

ITC methods in Java.lang.object :-

1. Public final void wait (long)
2. Public final void wait ()

} Throws Java.lang.InterruptedException

3. Public final void notify ()
4. public final void notifyAll ()

- Method-1 is used for making the thread to wait on the basis of some time.
- method-2 is used for making the thread to wait without time.
- Method-1 and 2 throws a predefine exception called Java.lang.InterruptedException.

- For more explanation see the sleep().

- Method-3 is used for transferring one thread at a time from waiting state to Ready state.
- Method-4 is used for transferring all threads from waiting state to Ready state.

- Classical Examples of ITC -

1. Producer-Consumer Problem.
2. Readers-Writers Problem.
3. Dining-Philosopher problem.
4. Barber-shop problem.

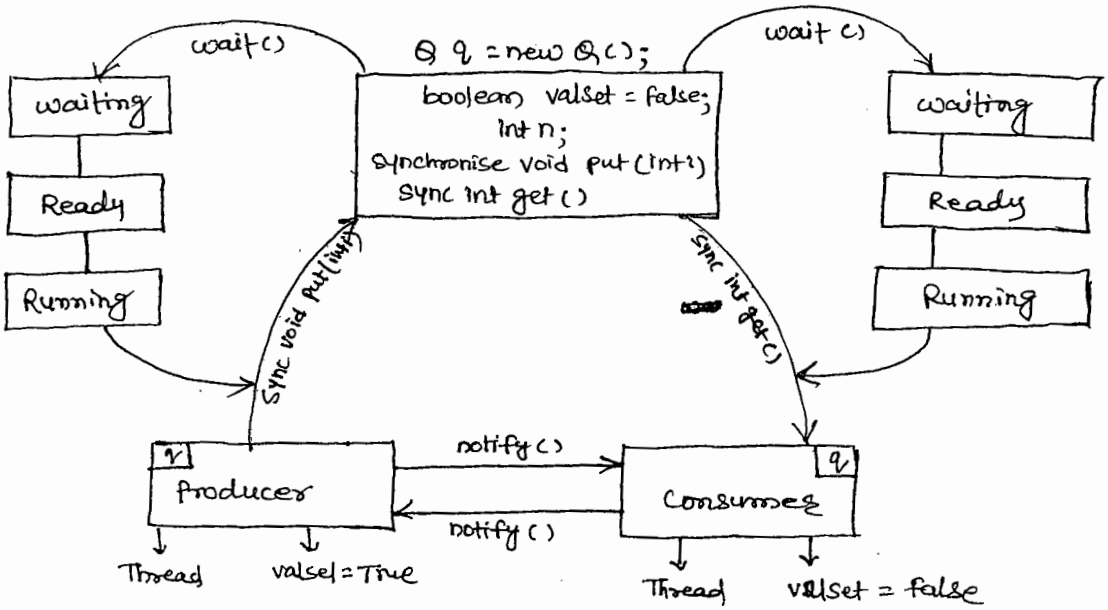
SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

• Implementation of Producer-Consumer Problem -

Producer - producer is one of the thread and whose role is to produce item/value which are consume by consumer. At any point of time producer allows us to produce only one item but not multiple items.

Consumer - consumer is also a thread, which is used for consuming the values, which are produced by producer. At any point of time consumer can consume only one item at a time but not multiple item.

The following diagram gives the implementation of Producer Consumer Problem -



SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

String Handling

- The purpose of String handling is that to store string data & to perform various operations on string data such as obtaining a character, reverse of string, inserting, deleting, modefining etc.

- Def. of character:-
A character is an Identifier enclosed within single quotes.

ex- 'A' '$' 'g' '20' .... etc — ✓
'Java' — x
'Java prog' — x

Def. of string:-
collection of character enclosed with double quotes; is known as String.

ex - "Java" ✓
"Java prog" ✓
"A" } ✓
"g" }
"$" }

In order to store string data & to perform various operation on string data.

we have four types of class —

a.) Java.lang.String
b.) Java.lang.StringBuffer
c.) Java.lang.StringBuilder
d.) Java.util.StringTokenizer

Differences between String & StringBuffer class —

String	StringBuffer
1. By default "__" is treated by JVM as an object of Java.Lang.String.	a.) By default "__" is not created by JVM as an object of StringBuffer.
2. String Buffer object initalization is possible without constructor. ex- string s = "Java";	b.) StringBuffer object initialization is not possible without constructor ex- StringBuffer sb = "Java"; //invalid

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

String	StringBuffer.
3.) when we create an object of String class additional memory is not created.	c.) when we create an object of StringBuffer we get 16 additional char. memory space.
4.) All string class object within the same memory space; modifications not permitted. (modifications permitted provided existing memory space is destroid & new data will placed in	d.) on StringBuffer obj modification can be permitted.
5.) String class obj is known as immutable obj.	e.) StringBuffer class obj is also known as mutable obj.

Similarity

- These two classes are belongs to public final.
- These classes never participates in inheritence.
- we can't override methods of these classes.
  (programmatically they never participate in IS_A relationship they participate always in has_A & uses_A relationship)

Profile of Java.lang.String

- Constructor :-

1. String () :- It is used for empty string object.

   ex- String s = new String ();

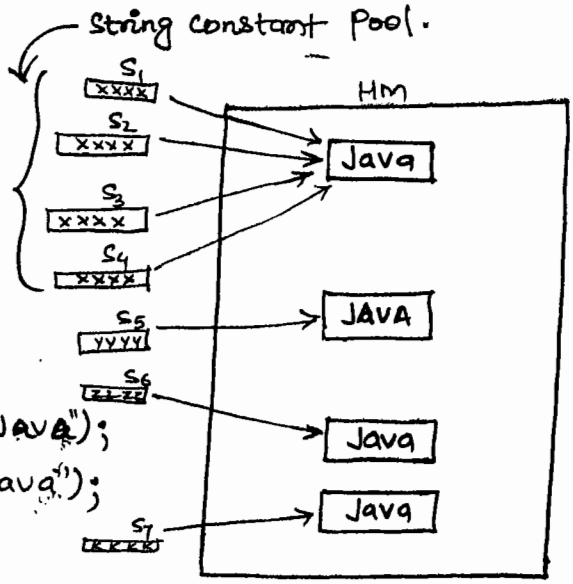
   SOP (s); // empty (nothing will print)

2. String (String):-

   It is one of the object parametrized constructor & it is used to copy the content of one string object into another string object.

ex-

```
String s1 = "Java";
String s2 = "Java";
String s3 = "Java";
String s4 = "Java";
String s5 = "JAVA";
String s6 = new String("JAVA");
String s7 = new String("Java");
```



* String constant pool :-

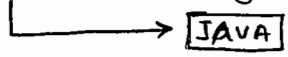
If we create multiple string objects by placing same data with same case & meaning without using constructor then the multiple string class objects are called string constant pool.

3.) String (char[]) -

This constructor is used for converting array of characters into string.

ex-

```
char ch[] = { 'J', 'A', 'V', 'A' };
String s = new String(ch);
```




```
SOP(s); // JAVA
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

4.) String (StringBuffer) :-

It is used to convert StringBuffer object into String class objects. In otherwords we are converting mutable object into immutable object.

ex:- StringBuffer sb = new StringBuffer("Java");
String s = new String(sb);

5.) String (StringBuilder) :-

It is used to converting StringBuilder obj. into String class objects.

Instance Method

1. **Public int length :-**

This method is used for finding the No. of character present in string.

ex-
```
String s = "JAVA";
SOP(s); //JAVA
int noc = s.length();
SOP(noc); //4
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

ex-
```
int noc = "Sathya".length();
SOP(noc); //6
```

ex-
```
String crs[] = { "CPP", "JAVA", "oracle" };
                  0       1       2
SOP("No. of cources:" + crs.length); //3
int noc = crs[1].length();
SOP(noc); //4
```

Note :- - 'length' attribute to be applied on arrays.
- length[] to be applied on String object but not be applied on opposite.

2. **Public char charAt(int) :-**

This method is used for finding a character by passing valid position otherwise we get predefine unchecked exception called java.lang.StringIndexOutOfBoundsException.

ex-
```
String s = "JAVA";
           0123
char ch = c.charAt(2);   // valid position
SOP(ch); //V
char ch = s.charAt(20);  // Invalid position
```
throws → StringIndexOutOfBoundsException

3. <u>Public char[] toCharArray()</u>

This method is used for converting string data into character Array.

ex– String S = "JAVA";

```
SOP(S); // JAVA
Char ch[] = s.toCharArray();
For (int i=0; i<ch.length; i++)
{
    SOP(char[i]); // JAVA
}
```

4. <u>Public String Trim()</u>

This method is used for eliminating leading & trailing spaces but not the spaces which are present between the words.

ex–

String pwd = " ␣␣␣ JAVA ␣ PROG ␣␣␣";

leading spaces — Trailing spaces Yeilding

```
int noc = pwd.length();
SOP(noc); //16
String ts = pwd.trim();
int noci = ts.length();
SOP(noci); // 9
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

5. <u>public String toUpperCase()</u>
<u>public String toLowerCase()</u>

The above methods are used for converting lower case data into uppercase & uppercase data into lowercase respectively.

<u>ex</u>:- String os = "JaVA";

```
String ls = os.toLowerCase();
SOP(ls); // java
String US = ls.toUpperCase();
SOP(US); // JAVA
```

6. ⓐ Public String subString (int)

This method is used for obtaining a part of the string by passing the position where to start as it takes up to last character.

ⓑ Public String subString (int s, int e-1)

This method is used for obtaining the range of character by specifying the position where to start (s) & where to end (e-1). i.e. (S to e-1) characters as a substring.

ex — String s = "JAVA PROG";
(0 1 2 3 4 5 6 7 8)

— String ss1 = s.subString (5);
sop (ss1); // PROG

— String ss2 = s.subString (2,7); // 2 to (7-1)
sop (ss2); // VA PR

7. ⓐ Public boolean equals (string)

This method is used for comparing two string contains this method returns true provided the content of source string & destinating string must be same in case & meaning otherwise it returns false.

ⓑ Public boolean equalsIgnoreCase (string) :-

This method return true provided the content of source string & destination string are same in meaning without considering the

This method returns false the contents of source string & destination string are different in meaning.

ex- String s1 = "JAVA";
String s2 = "java";
boolean b = s1.equals(s2);
sop (b); // false
b = s1.equals ("JAVA");
sop (b); // true

```
b = s1.equalsIgnoreCase(s2);
    sop(b); // True
b = s.equalsIgnoreCase("KAVA");
    sop(b); // False
```

Q. what is difference between equals() & "==" operator.

- equals() is always used for comparison of contents of two String objects; where as "==" is used for comparing references of two objects.

ex -

```
String s1 = "JAVA"
String s2 = "JAVA"
String s3 = new String("JAVA");
String s4 = new String(s2);
if (s1 == s2)
{
sop("s1 & s2 contains same ref.");  // valid
}
else
{
sop("s1 & s2 contains Diff. ref.");
}
if (s2 == s4)
{
sop("s2 & s4 contains same ref");
}
else
{
sop("s2 & s4 contains diff. ref");  // valid
}
if (s1.equals(s2))
{
sop("same content");  // valid
}
else
{
sop("diff. content");
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

8. Public boolean startsWith (String) :-
Public boolean endsWith (String) :-

- The above methods are used for pattern matching.
- This method returns true provided the destination string starts with / ends with source string respectively otherwise they returns false.

```
ex- String s = "JAVA is a OOP Lang";
    boolean b = s.startsWith("JAVA");
            sop(b); // true
    b = s.startsWith("Lang");
            sop(b); // false
    b = s.endsWith("JAVA");
            sop(b); // False
    b = s.endsWith("Lang");
            sop(b); // True
```

9. Public String concat (String) :-

This method is used for concatinating contents of two strings but not more than two strings contents.
To do more than two strings contents concatenation we use "+" operator.

```
ex- String s1 = "Sathya";
    String s2 = "Tech";
    String s3 = s1.concat(s2);
            sop(s1); // Sathya
            sop(s2); // Tech
            sop(s3); // sathya Tech

ex:- String s3 = s1 + s2 + "Hyd";
            sop(s3); // Sathya Tech Hyd
```

10. Public int compareTo (String)

This method returns either zero, Positive or negative value.

ex:- String S_1 = "JAVA";
String S_2 = "java";
int cv = S_1.compareTo ("JAVA");
sop (cv); // 0 ⇒ equals

cv = S_1.compareTo (S_2);
sop (cv); // -ve ⇒ Desti. string (S_2) is greater than source string (S_1).

cv = S_1.compareTo ("ABC");
sop (cv); // +ve ⇒ Source String (S_1) is greater than Destination String (ABC)

* Static Method:

Public Static String Value of(xxx)

In this method xxx represents any fundamental data type; it is one of the overloaded static factory method & it is used for converting fundamental type values into String type values.

ex:- → int a = 10;
String iv = String.valueof (a);
sop (iv); // 10

"10"

→ float x = 1.5f
String fv = String.value of (x);
sop (fv); // 1.5

"1.5f"

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

StringBuffer (Java.lang.StringBuffer)

- It is one of the predefine public final class and it never participates in inheritance process.
- when we create an object of string buffer we get 16 additional character memory space and whose object is known as mutable obj.
- To perform the operation on string data by using StringBuffer we need to learn its profile.

Profile of Java.lang.StringBuffer :-

Constructor:-

1. StringBuffer(char [])
2. StringBuffer(StringBuffer)
3. StringBuffer(String)
4. StringBuffer(StringBuilder)

- constructor ① is used for converting array of characters into String Buffer object.

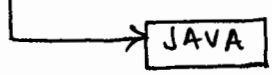
```
ex- char ch[] = { "J", "A", "V", "A" };
    StringBuffer sb = new StringBuffer(ch);
    sb ---> [ JAVA | 16 add. character ]
    SOP (sb); // JAVA
```

- constructor ② is used for converting one string buffer into another String Buffer object.

```
ex- StringBuffer sb = new StringBuffer ("JAVA");
    StringBuffer sb1 = new StringBuffer (sb);
    SOP (sb);   // JAVA
    SOP (sb1);  // JAVA
```

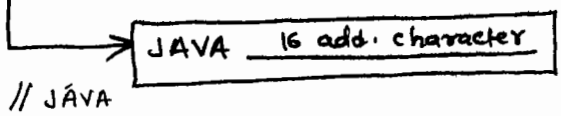
- constructor ③ is used for converting String class object data into StringBuffer object.
  in other word this constructor converts immutable object into mutable object.

ex:- String s = "JAVA";



SOP (s); // JAVA

StringBuffer sb = new StringBuffer (s);



SOP (s); // JAVA
SOP(sb); // JAVA

- Constructor ④ is used for converting StringBuilder object into StringBuffer Class.

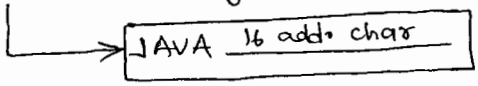
Note:- String Constant pool exists only for String class object because one can initialize the ~~class~~ String class object without new operator where as string constal pool not exists for StringBuffer class object because every time we place the data with new operator but not directly.

Instance Method:-

- public int length().
- public int capacity()

- The above methods are used for finding no. of characters and capacity. (no. of character + 16 add. character) of StringBuffer object.

ex:- StringBuffer sb = new StringBuffer ("JAVA");



SOP ("No. of char =" + sb.length()); // 4

SOP ("capacity of sb =" + sb.capacity()); // 20 (16 + 4)

- public StringBuffer reverse()

This method is used for obtaining sentence ~~result~~ reverse of StringBuffer object.

ex:- StringBuffer sb = new StringBuffer ("JAVA PROG");

StringBuffer sb1 = sb.reverse();

SOP (sb); // JAVA PROG

SOP (sb1); // GORP AVAJ

Public String toString()

- This method is used for converting StringBuffer object (mutable) into StringClass object (immutable).

ex:-
```
StringBuffer sb = new StringBuffer("JAVA");
String s = sb.toString();
sop(s); //JAVA
```

- Public void trimToSize()

- This method is used for eliminating Additional character memory space and adjust the String Buffer object memory only to it length (upto).

ex:-
```
StringBuffer sb = new StringBuffer("JAVA");
sop("cap. of sb =" + sb.capacity()); //20
sb.trimToSize();
sop("cap of sb after trim =" + sb.capacity()); //4
```

- Public int codePointAt(char):-

This method is used for obtaining unicode value of specific character which is present in the object of StringBuffer.

ex:-
```
StringBuffer sb = new StringBuffer("JAVA");
int uc = sb.codePointAt('A');
sop("uc"); //65
```

- public void setCharAt(int, char) -

This method is used for replacing the existing character with new character by specifying the position of existing character.

```
StringBuffer sb = new StringBuffer("KAVA");
sop(sb); // KAVA (0 1 2 3)
sb.setCharAt(0, 'J');
sop(sb); //JAVA
```

- public StringBuffer append(Object)
- Public StringBuffer append(String)
- public StringBuffer append(StringBuffer)
- public StringBuffer append(XXX) // Here XXX is fundamental data type

- The above methods are used for appending String data, StringBuffer data, fundamental type data or any type of object data to the content of existing StringBuffer data.

ex:-

```
StringBuffer sb = new StringBuffer ("JAVA");
StringBuffer sb1 = sb.append ("PROG");
        SOP( sb1); // JAVAPROG
StringBuffer sb2 = sb1.append (10);
        SOP ( sb2); // JAVAPROG10
```

Public StringBuffer delete (int, int) —

This method is used for removing range of characters by specifying the position where to start & where to end.

ex:-

```
StringBuffer sb = new StringBuffer ("JAVAPROG");
        SOP(sb); // JAVAPROG
StringBuffer sb1 = sb.delete (4,7);
        SOP (sb1); // JAVA
StringBuffer sb2 = sb1.deleteCharAt (3);
        SOP (sb2); // JAV
```

Public StringBuffer deleteCharAt (int) —

This method is used for deleting a particular character by specifying its position.

Public StringBuffer replace (int, int, String) —

This method is used for replacing range of character with the specified new string.

ex:-

```
StringBuffer sb = new StringBuffer ("JAVAPROG");
        SOP (sb); // JAVAPROG
StringBuffer sb1 = sb.replace (4,7, "LANG");
        SOP (sb1); // JAVALANG
```

Public StringBuffer insert (int, String) —

This method is used for inserting a new string in the existing data of StringBuffer; by specifying the existing position.

```
StringBuffer sb = new StringBuffer ("JAVALANG");
        SOP (sb);
StringBuffer sb1 = sb.insert (4, "PROG");
        SOP(sb1); // JAVAPROGLANG
```

String Builder

Java.Lang.StringBuilder:-

The functionality of StringBuilder is exactly similar to StringBuffer but StringBuider methods are Non-synchronize gives non-Thread safety result and recommended to used in server-side-Appl.

where as StringBuffer class object and whose method are synchronized and gives thread safety result and recommended to used in client-side-Appl.

- Constructor of StringBuilder are similar to StringBuffer, methods of StringBuilder are similar to StringBuffer without synchronized.

StringTokenizer

Java.util.StringTokenizer -

- It is the one of the predefine class present in Java.util.*
- The purpose of StringTokenizer is split the given String into different substring based on delimeter.
- Some of the delimeter are . , ; :
- String Tokenizer always seperate the given String by considering delimeter. delimeter seperated values are called Tokens.

Profile of Java.util.StringTokenizer

Constructor

1. StringTokenizer (String)
2. StringTokenizer (String, String)

Instance Method-

3. Public int countToken();
4. Public boolean hasMoreTokens();
5. Public String nextToken();

- constructor ① is used for creating an object of StringTokenizer without using object specifying.
- constructor ② is used for creating an object of StringTokenizer by specifying delimeter.
- method ③ is used for counting the no. of Tokens which are seperated by delimeter.
- method ④ returns true provided an object of StringTokenizer having more no's of Tokens otherwise it returns false.
- method ⑤ is used for obtaining next token w.r.t. the object of StringTokenizer provided Method ④ returns true.

Q. write a Java Program which illustrate the Concept of StringTokenizer.

```
// StrTok.Java
import Java.util.StringTokenizer;
class StrTok
{
 PSVM (String [] args)
 {
  StringTokenizer st = new StringTokenizer ("Java # is #
                                  on #Long", "#");
   int not = st.countTokens();
   SOP ("no of Token ="+not);
   while (st.hasMoreTokens()).
   {
     String tok = st.nextToken();
      SOP (tok);
   }
 }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Generics

- It is one of the new Facility present in JDK 1.5 version.
- Generics concept is recommended to apply on overloaded concepts like overloaded method and overloaded constructor.
- In other words Generics concept can be applied in the following cases —

Case-I

Let us assume there exist family of similar overloaded method calls. In the ordinary programming in Java for each & every overloaded method calls we must define a seperate method definition. This approach leads to -

(a) more Appl^n development time.

(b) Redundancy of the code is more.

(c) performance is degraded.

To overcome above problems we apply the concept of Generics on family of overloaded method calls and results in single method definition and it is called Generic method.

Definition of Generic (-) Method —

A Generic (-) is one whose definition exist only one's irrespective of family of overloaded method calls.

Case-II

Let us assume there exist Family of overloaded constructor calls. with the ordinary programming of Java for each and every overloaded constructor we must define a seperate constructor definition. and leads to -

- more Appl^n dev. time
- Redundancy of the code is more
- Performance is degraded

To overcome above problem we apply the concept of Generics on family of overloaded constructor definition and results in single constructor definition; and it is called Generic constructor

Def. of Generic constructor:—

A Generic constructor is one whose definition exist only once irrespective of family of overloaded constructor calls.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

case-III

Let us assume there exist a class definition with two int datamember and if we create an object of that class then in that object we can enter only int values but not other type of values. If we want to enter other type of values in the same object then it is not possible. To make it possible, with ordinary programming we have to define Multiple class definition with different type of DataMembers. This approach leads to –

- More Appl^n dev. time.
- Redundancy of code is more.
- performance is degraded.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

To overcome these problems, we apply the concept of Generic on multiple class definition which are containing different data member and it results in generic class.

definition of Generic class.

A Generic class is one whose definition exist only once. irrespective of no. of object created with different type of value with the same class.

Adv. of Generics –

1. Application development time is less.
2. Redundancy of the code is minimized.
3. performance of the application is inhanced.

- Generics concept makes the programmer to ensure either overloaded method or overloaded constructor must have same no. of Parameter with the different types and order.

- Generic concept always deals with objects (it facilatates the programmer to supply the data in the form of fundamental type and obtains [illegible] type.).

SRI RAGHAVENDRA XEROX

Syntax for Generic class, Method and Constructor :-

```
Class <clsname> <Gen var1, Gen var2 .... Gen varn>
{
Var Declaration belongs to Gen Var1, Gen var2, .....Gen varn ;
<cls name> ( List of formal Parameter belongs to Gen var1, ...Gen varn
                                                  if any )
{
  Block of stat(s) .... Initialization;
}
Ret. Type  method Name ( List of formal Parameter belongs to Genvar1...Genvarn
                                                  if any )
{
  Block of stat(s) ... BLG
}
}
```

Generic constructor

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road.

- Here Genvar1, ... Genvarn represents Java valid variable names and they always contains types.
- ReturnType represent type of value being returned by a method. If a method is not returning any value then it return type must be void.
- If method is returning some values then its return type must be Generic type.

Syntax for creating Generic Object :-

```
<clsname> <cls1, cls2 ..... clsn> <obj.name> = new <cls name>
                                                <cls1 .... clsn> ( List of values )
```

Here <cls name> ⇒ Generic class

<cls1, cls2, ... clsn> ⇒ wrapper class or ~~wrapper~~ Non-wrapper class Name

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. WAJP which illustrate the concept of generic class, generic method, generic constructor ?

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Design Pattern

- The purpose of design pattern is that to provide robust, more durability, less Maintainence.

def:-

Design patterns are the best rules invented and prouved by industry experts and realease to real industry used by the programmers to eliminate Recuring problem occuring in s/w development.

- In real-world we have many no. of design Pattern as a Part of JSE, JME, JEE.

As a Part of JSE we have two types of design Pattern they are:-

(a) Factory method
(b) Singlton class

(a) Factory Method

A Factory method is one whose return type is similar to class name in which class it present.

The purpose of Factory method is creating an object of any class without using "new" operator.

ex:- (a) Java.awt.GraphicsEnvironment

Public Static GraphicsEnvironment getLocalGraphicsEnvironment()
(3) (2) (1)

(b.) Java.lang.String

Public Static String valueOf(xxx)
(3) (2) (1)

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

rules for developing Factory Method:-

1. The returntype of factory method is similar to class Name in which it present in class.
2. The Nature of every Factory method must be static.
3. The Access modifier of the factory method is recommended to Public

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

(b) Singleton class

singleton is one of the class in Java which allows us to create only one object Per JVM.

Steps/guideline for designing Singleton Class

1. choose an appropriate package Name for placing singleton class for common access and ensure that package statement must be first executable statement.

2. Choose an appropriate class Name and ensure that whose modifier must be public (this class is going to be called as singleton class).

3. E & E singleton class must contain self-declared object (contains Null) with private static nature.

4. E & E singleton class must contain explicit private default constructor.

5. E & E singleton class must contain a factory method by satisfying its rules.

6. whatever the singleton class we are placing in the package whose name must be given as filename with an extension .Java

Q: Define a class which will implement singleton class rule?

```
// Sathya.Java ———— (6)
Package sp; // ———— (1)
public class sathya // ———— (2)
{
  Private static sathya st = null; // ———— (3)
  Private sathya () // ———— (4)
  {
    SOP ("object Created First Time");
  }
  Public static sathya create () // ———— (5)
  {
    if (st == null)
    {
      st = new sathya ();
    }
```

```
else
 {
  sop ("object already created");
 }
 return st;
}
```

Compile:→ `Javac -d . Sathya.Java`

Q. write a Java main program which makes use of sp.sathya

```
// DepPatDemo.Java
import sp.sathya;
class DepPatDemo
{
  psvm (string [] args)
  {
    //sathya s = new sathya(); invalid bcoz sathya() has private
                                 access

    Sathya s1 = Sathya.create();
    Sathya s2 = Sathya.create();
    Sathya s3 = Sathya.Create();
  if ((s1 == s2) && (s2 == s3) & (s3 == s1))
   {
    sop ("multiple object - same reference - one object");
   }
   else
    {
    sop ("multiple object - different reference - diff. obj");
    }
  }
}
```

- Here sp.sathya class is allowing thus to create single obj for JVM and hence it is called singleton class.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Inner Class / Nested Classes

- If we define a class 'Y' inside the definition of another class called 'X' then the class 'Y' is called inner class and class 'X' is called outer class.

Definition of containership :-

The process of defining one class definition into another class definition is called containership

We have four types of inner classes :-

1. Static inner class
2. Member inner class
3. Local inner class
4. Anounymous inner class

1. Static inner class :-

The purpose of static inner class is performing one-time common operations by taking static features.

- Just static classes are not there in Java, but one can write inner static classes.
- we can't refer outerclass Instance properties (Data members, methods) from the context of inner static class.
- Programatically, in the static inner class we can take both instance & static features. (recommended to take static features)

Syntax :-

```
class <outer class>
{
  ____
  ____
  static class <inner class>
  {
    ____
    ____
  } //inner-class
} //outer class
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

Q. WAJP which illustrate the concept of static inner classes:

```
//XY.java

class X
{
 int a=10;
 static int b=20;
 void xyz()
 {
  sop("val of a of X="+a);
  sop("val of b of X"+b);
  sop("val of c of Y="+new Y().c);
  sop("val of d of Y="+Y.d);
 }
 static class Y
  {
   int c=30;
   static int d=40;
   void abc()
   {
    // sop("val of a from X="+a);      invalid
    sop("val of b from X="+b);
    sop("val of c from Y="+c);
    sop("val of d from Y="+d);
   }
  } Y-inner class

} X-outer class
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
class XYZ
{
 psvm (String [] args)
 {
   X O1 = new X();  // creating an object of outer class
   O1.XYZ();
   X.Y O2 = new X.Y();  // creating an object of static inner class.
     O2.abc();
 }
}
```

2. <u>Instance Inner Class</u> :-

- Instance inner classes contains only instance features and it never contains static features.
- The purpose of instance inner class is performing some repeated operation by containing instance features.
- Instance inner class can access all features of outer class and outer class can also access instance inner class feature.

Q. WAJP which illustrate the concept of instance inner class.

```
// PQ.Java
class P
{
  int a=10;
  static int b=20;
  void abc()
  {
    sop("val of a of P = "+a);
    sop("val of b of P="+b);
    sop("val of c of Q="+new Q().c);
  }
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

```
class Q
{
  int c =30;
// static int d =20;    — invalid
// static void pqr () { }  —— invalid

void xyz
 {
   sop(" val of a of P =" +a);
   sop (" val of b of P=" +b);
   sop (" val of c of Q" +c);
 }//xyz ()
}// Q - instance inner class
}// P - outer class
class PQ
{
  P s v m (string [ ] args)
   {
     P o1 = new P ();
     o1.abc ( );
     P.Q o2 = new P().new Q (); // creating an obj of
                                   instance inner classes
     o2.xyz ();
   }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

3. Local Inner Class –

- If we define a class definition inside the method of another class than class definition inside the method definition by called Local inner class.

- Local inner class must be instance class but it shouldn't belongs to public, Protected, Private and static.

- Local inner class methods can be accessed within the containing method only but not in other methods of outer class.

Q. WAJP which illustrate Local inner class ?

```
//LIC.java
class Sathya
{
  void LearnJava()
  {
    SOP ("Learning Java from Sathya");
    class Faculty
    {
      void teachJava()
      {
        SOP ("learning from xyz faculty");
      }
    } // Faculty - Local inner class
        faculty F = new Faculty();
        f.teachJava();
  }//learnJava()
} // Sathya - outer class
class LIC
{
  PSVM (String [] args)
  {
    Sathya S = new Sathya();
        S.learnJava();
  }
}
```

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

4. Anonymous Inner Classes

- An anonymous inner class doesn't contains any .

- The purpose of anonymous inner classes is to implement abstract classes and interfaces.

- For more explanation & more examples see inner interfaces (in previous pages before package topics)

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.

SRI RAGHAVENDRA XEROX
Software Languages Material Available
Beside Bangalore Ayyagar Bakery,
Opp. CDAC, Balkampet Road,
Ameerpet, Hyderabad.