(6) JOINS

- Inner
- left
- Right
- full join

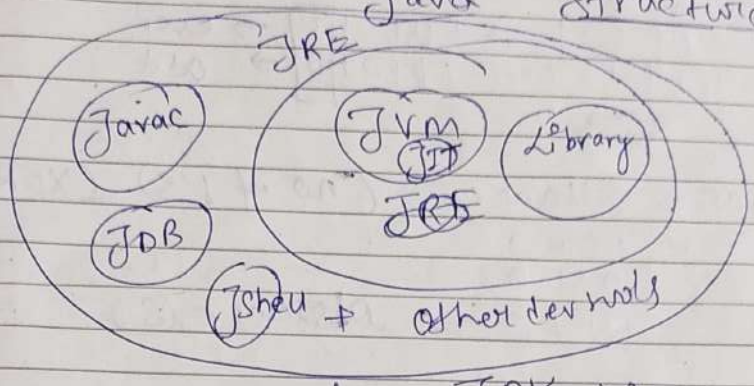
select *
from table A
left join table
on tableA.id = tableB.id.

28 → 3 jun

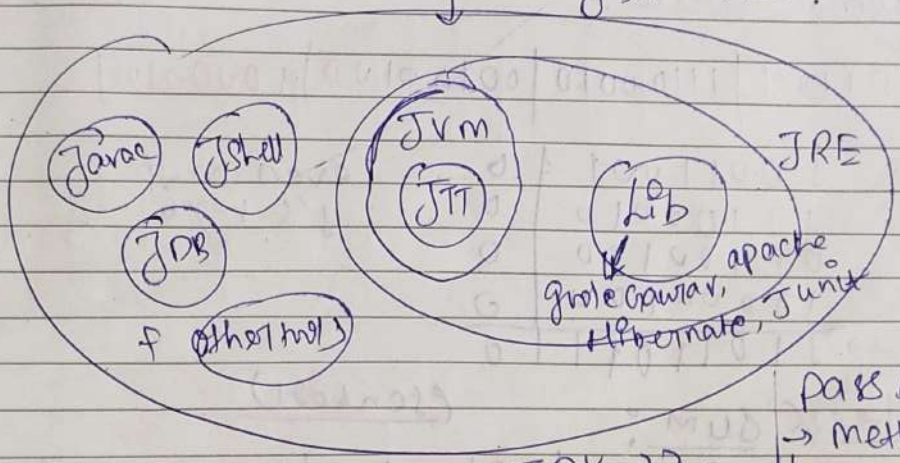
11:10 - 12:00

3.30 - 4.30

Java structure



JDK - 22.



JDK - 22.

* final, finally, finalized ?

pass by value :-

→ method is called by passing actual value of argument, value of argument will not change.

pass by reference

(17) Inheritance [Inherit methods & variables of superclass]

Single (A→B) | Multiple (C→A, C→B) | Multilevel (B→C, B→A) | Hierarchical (A→B, A→C) | Hybrid (A→B, A→C, B→D)

(18) polymorphism :

runtime / method overriding

compile time / method overloading

~~call by value~~ ~~call by reference~~

(19) collection framework
(1) Set (2) Hashmap (3) Arraylist (4) Queue (5) Stack (6) array :

(20)

Process	Thread
* A process is a program in execution	* Basically a block of sequence of instruction
* More time to terminate	* less time to terminate
* process is isolated	* shared memory
* & shared data with others.	* shared data with oth

(21) Life cycle of thread.

new → Runnable → Blocked → Waiting

Terminated

(21) applet :-
It is a java program which embedded into webpage. It's run inside browser and work as a client side.

init() → start() → paint() → stop() → destroy()

applet execution.

(22) awt → application window toolkit.
swing → advance version of awt. It is platform independent wise more library func use dialog box.

(9) Array is created in heap memory.

(10) Base Address + Index * no-of-bytes.

(11) Concepts of oops → (i) Inheritance
(ii) Polymorphism
(iii) Abstraction
(iv) Encapsulation.

(12) Java Script → Object-Based Programming
Java, C# → Object-Oriented

(13) Access specifier → (i) public
(ii) private
(iii) protected
(iv) default

Access from other class → (1) public
Access from same pkg → (2) protected
Access with subclass, same class, same package → (3) default
(4) private → Access within same package.

(14)

	Abstract class	Interface class.
*	Doesn't support multiple inheritance.	* Support multiple inheritance
*	Support final method	* not support final method
*	extend keyword is used	* Implement keyword is used always
*	method may be abstract or not	* method may not be abstract.

(Data hiding / code reusability)

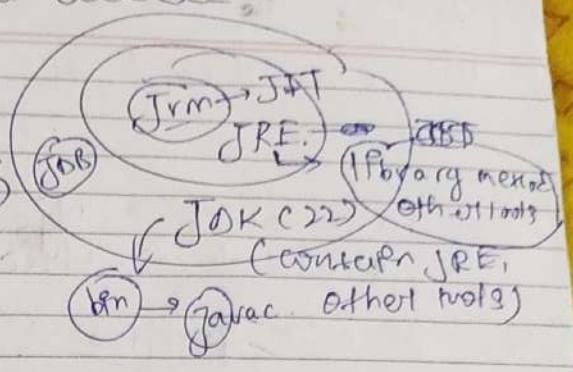
(15) Encapsulation: It is basic concept of oops in which variables & methods are bind together in a class and make instance variable as private so that they can only access within a class.

(16) 'IS-A' is type of rel" in OOP where one class inherits another class.

68 → keyword

(On-point wise) Java (Interviews)

1. Jvm executed byte code. Javac compile from byte code. (Source code)



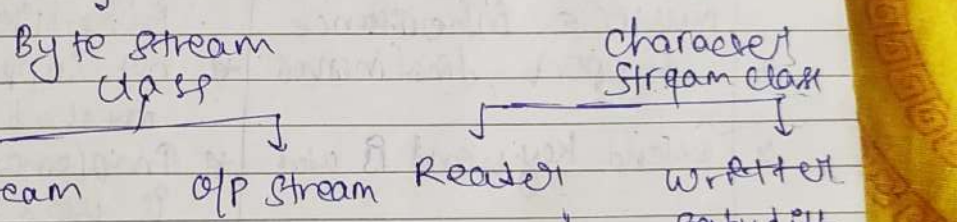
2. Robust
Secured
Multithreaded
Intangible
Object-oriented.

3. Storage memory: (I) Class (Method) Area (II) Heap (III) Stack (IV) PC register (V) method stack.

4. Classloader responsible for storing classes & Interface in Jvm.

5. Wrapper class → encapsulate the primitive Integer (Object class) data type.

6. Super class



7. Taking I/P from the console.
   1. Command line argument
   2. Buffear readder
   3. Concole → .console().readline()
   4. Scanner.

8. O → Operator:
   1. Arithmetic (+, -, *, /)
   2. unary (!, -, ++, ~)
   3. ternary (? :)
   4. Bitwise (|, ^, &, ~)
   5. Logical (&&, ||, !)
   6. Relational (==, !=, <, >, <=, >=)
   7. Shift (<<, >>, >>>)
   8. Instance of Oper.

Compaiso (Check Object reference).

→ by diff pkg, class

1. Public
2. protected → within same pkg, class (even in different pkg)
3. default
4. Private.

Bin search

int n = Arrays.sort(arr, n);

Public :- Accessible every where
protected :- Accessible within same pkg & sub class
Default :- Accessible within same pkg
private :- Accessible within same class.

Location of an element in 2-D matrix:-
A[j,k] = BaseAddress(A) + Bytes[Total rows*[k-1] + [j-1]].

(2)

1) Longest subarray consist distinct element diff K?

(1 2 2 3 3) 1 1 1 2 2 2 3 3 3 3 3

6 ... 6 1 1 - 7 + 1

int max = 1, end = 1   1 2 2 3 3   (8) (3)

st = 0;   0 1 2 3 4   3 11 - 3 = (8)

(end - 1) - 3

if (!map.containsKey(map.get(0))) {
map.put(map.get(0), 1);   Max =

while(end < arr.length) {
if (map.get(end) - map.get(end-1) ||
di(map.get(end) - map.get(end-1) == k) {
if (!map.contain(map.get(end))
map.put(map.get(end), 1)

else (map.replace(arr.get(end),
map.get(end).

se. {
map.clear();
map.put(arr.get(end), 1)
} st = end;
(map.st2 == 2) { max = Math.max(max, end - st + 1)
retu max;

→ Marker class :- Used to identify objects w/o adding functionality

→ Marker interface :- Use to provide metadata about a class to jvm, compiler w/o adding any functionality.

→ Frame work :- It is layered structured indicating what kind of program can be built.

→ Metadata :- [data about data] data providing info about one or more aspect of data.

→ final class or singleton class is same.

→ Navigableset may be accessed and traversed in either ascending or descending order.

→ Tree set :- Implement nevigable set keeping element sorted ord

→ Tree map :- Implement nevigable map & store key-value sorted.

→ collection-class :- Hashamap, ArrayList LinkedList, Stack, queue, HashTable

→ collection - Interface :- collection, set, queue dequeue.

→ Navigable map :- It is extension of sorted map

→ Handle Exception in java :-
① Try catch ② finally ③ throw & throws

→ Handle exception in both compile & runtime

→ Checked Exep :- ① file not found, Sql, Io (checked) (unchecked)

→ un checked :- Null pointer, Arithmetic, Array out of Bound.

* Foreign can't contain NULL value.
* Primary key can't contain NULL value.

→ Constraints :- Are the rules which enforce integrity, consistency, security. common types are not null, primary key, foreign key. also help to ensure valid data enter into data base.

→ Redundancy :- occur when same data is stored into a multiple place that refer storage wastage & inconsistency. solve using normalization.

→ Pckg :- In java it is a namespace that organize related classes and interface. use for grouping of related classes, avoid name conflict, security & reusability.

→ 53 Keyword in java.

→ JCF :- [Java collection Framework] :-
→ powerful tool in java for efficient data handling and also provide a wide range of interface & classes. to meet various data storage.

→ Collection :- Root interface for all collections
→ Threading :- ~~Allows~~ Java ~~to~~ java allows program to perform multiple task by dividing ~~single thread into unit~~ a program into unit called thread.

→ Java doesn't support pass by reference.
→ Memory management :- It is process of controlling and coordinating ~~and~~ memory including memory allocation & deallocation to a program.
→ GC (Garage collection) is an automated memory management features in java, Python

(B.) Polymorp: eg:- sound method behave differently for dog & cat class.

Allows methods to behave differently based on the object called them.

(4) Abstraction :- Hides complex implementation details, showing only essential part.

eg:- abs class shape that initialize abs method areaCS and that is declare in circle class.

→ JRE :- Installation pckg that provide env to run java program

JVM Platform dependent.

→ JDK :- provide env to develop & execute (pckg) Java program.

→ dot operator :- used to access classes & sub-packages from the package.

→ library & pck & their requirement?

→ Contructor :- It is called when an object of a class is created.

→ library → is a group of pckg.

→ Pckg in java essential for organizing classes, interface & sub pakge into structured format ; to make code reuse & avoid naming conflict

→

→ Java lib :-

→ Apache common [use for reusing java component]

→ Google Gauava [And utilities].

↳ set core lib that include collection, caching, concurrency & string manipul.

→ Junit [Framework for writing & running unit test]

→ Hibernate [ORM Object Relation Mapping use for database interaction.

Java	Python
(1) Syntax uses {} & semicolon	(1) use Indentation.
(2) Faster	(2) Slower.
(3) uses both compiler & Interpreter (JVM, Javac)	(3) use Interpreter Python.
(4) Handle exception during compile & runtime	(4) Handle exception during runtime.
(5) use in web dev, data scien, ML	(5) use in web-App, android dev
(6) ~~less secure~~	(6) ~~more secure~~.

Q. Abstract constructor in java. [We can create constructor of abstract class but abstract key can't be use ~~in~~ with constructor.

→ constructor can be final & non-final/final class.

→ We can create constructor of final class but final key not use in constructor.

Q. → ~~Destructor~~ Destructor ? [C++]

⇒ JIT [Just In Time] part of JVM which is used convert byte to native m/c code.

(by compiler) → Abstraction :-

(1) Encapsulation :- basic concept of oops that bind variables & method together in side class and making instance of class private so that it can accessible within same class

eg :- Person class Inherit name & age.

(2) Inheritance :- It also concept of oops which allow child to inherit method & varia ccess. of parent class :- animal inherit by dog & cat

while	do-while
(1.) Entry control loop	(1.) exit control loop
(2.) condition check first then exec	(2.) Execute first the cond^n is checked
(3.) ~~If~~ No statement execute if cond^n is false	(3.) Statement execute at least once if cod^n is false

PL/SQL	SQL
(1.) Procedural language	(1.) Not PL
(2.) follow procedural framework	(2.) follow basic operat CRUD.
(3.) complete programming langue	(3.) ~~Not~~ query-langu

→ ~~TCL~~ TCL → Rollback [undo last cmnd not save yet], commit [save all commands], save point [set a point upto which all cmnds saved]

→ Drop → delete whole scheme
→ Delet → comes up with cod^n that del specific row
→ trunc → use to delete all tuples.

→ Constructor In Java :- [can be privat [can't be inherit] / Can't override / ~~be~~ ove
(a) Default :- [no-argument automatically provi
(b) No-argument :- [no parameter]
(c) Parameterized :- [parameter]
(d) Copy constructor :- [creates a new object by copying the values of an existing obj]
(e) constructor can't override but can overload.

View :- It is a virtual table contain rows & colum like data table and contain defined by a query

Trigger :- It is a object in database which is fired automatically when insert, deletion, cre

Linear	Non-linear
1. Data are stored in sequential way & order manner.	1. Data are stored in hierarchical form.
2. Traversal perform in one go	2. Traversal perform level by level.
3. memory inefficient	3. memory efficient
4. easy to implement	4. ~~easy~~ hard to implement
eg:- Array, Stack, Queue	eg:- Tree, graph, heap.

→ level order :- Ascending order
→ Postorder :- L - Right → Root
→ Pre order :- ROOT → left → Right
→ In order :- left → Root → Right.

→ POST → Inorder → Preorder.

→ HashMap & HashTable :-

N: In Java, HashMap & Hashtable collective consent. are both key-value storage class that are using hashing techniques to store unique keys.

→ The main difference is HashMap is not synchronized by default while hashtable is synchronized.

(ass)
→ HashMap allows one null key & multiple null value which hashtable doesn't allow any null keys or values.

$h(k) = (K) \% (n)$ → no of buckets
↓ size of table.
Hash fn:
key

Array	ArrayList
① Fast	① slower due to resize overhead & maintaining internal a
② ~~Slower~~ due to contiguous block of memory	② In-built method
③ no in-built method	③ Dynamic size.
④ Fixed size	④ Object / wrapper class.
⑤ Primitive datatype	

* final → Prohibit modification & overriding
* finally → Ensures code run after 'try-catch' (use for clean up)
* finalize():
  ↳ called by the garbage collector before object destruction; typically use for clean up. (resources like: file handle, database, network connection)
* throw :- Manually throws an exception.
  use inside method. a method
* throws :- Declare exception that might throw
* ArrayList :-
  → Accept instance of a wrapper class
  → Don't store address.
  → Resize overhead
  → O(1) for index-based access [Access time
* Linked List :-
  → Don't accept instance of wrapper class
  → Store address of next node
  → NO resize overhead. (adjust size w/o reallocation
  → O(n) for access time.
* class can be final in java & can't inherit by subclass, even if class is not final, method can be final. ~~final only; can final~~
* Instance use outside method, class or global var. when static key use befor variable, local variable use inside method.
* Final class can't inherit. ~~only once by final sub~~
*

String	StringBuilder	StringBuffer
Immutable	mutable	mutable
left efficient for modification	more efficient for modifical	Efficient but slower due to sync.
Thread safety (N/A)	Not thread safety	thread-safety
Hyped String	Single-thread modification	Multi-thread modification.

Array & ArrayList :-

Adapter pattern → Allows incompatible interface of different class to work together by wrapping one class to another class.

→ Decorator pattern → Allow adding new functionality & behaviour to an object dynamically by wrapping it with one or more decorator obj.

→ Observer & State.

→ Mvc & MVVM → Introducing Facade-pattern (new model).

~~Verbal & Reason~~

→ pass by value & pass by reference :

→ method is called by passing actual value of argument the value remain same.

→ method is called by passing object as a argument and reference of an object is passed by value, the value change later on.

→ Data base : Structured collection of organize data

→ DBMS : It is software which deals with data base, user and application to analyze data

→ DBMS : It is a software that manages databases, providing tools for data storage retrieval, security & integrity.

→ SQL :- Structure query language or Data Base programming language we to perform CRUD operations.

→ weak entity :- Entity which are not uniquely identifies by their own attribute and depend on strong entity. uniquely.

Step

→ strong entity :- Entity which are identifies by their own attribute and independent of other attribute.

select r = Num%m   r < (m+1)/2 → true → Num-r
for join   ↳ false → num+(m-r)
where H ——— x ———
group
having
orderlimit

Java Full Stack Engineer
= x = x = x =

* SOLID :- S O L I D → Dependency
↳ open closed → Interface Inversion
Single Responsibility Liskov Substitution Segregation

S → This principle state that class should only have one responsibility. Furthermore, it should ~~one responsibility~~ only have one reason to change

O → classes should be open for extension but closed for modification. In doing so, we stop ourselves from modifying existing code and causing potential bugs.

L → If class A is a substype of class B, we should be able to replace B with A without disrupting the behavior of our program.

I → larger interface should be split into smaller ones. By doing so, we can ensure that implementing classes only need to be concerned about the methods that are of interest of them.

D → refer to decoupling of software, instead of high-level module depends on low-level module both are depends on abstraction.

→ Instance of a class = obj of a class.
→ this keyword = instance variable.

→ Singleton-pattern → one instance of a class.
→ Builder-pattern → way to construct complex object
→ Factory-pattern → Provide interface to create obj but allow subclass to decide which class need to be instantiated

E R Model

① Entity weak/strong
② Attribute: key, multivalue, composite, descr
③ Relationship: ① one-one ② many one ③ one many ④ many many

E-R Symbol:
① ellipse, ② rectang (strong entity) ③ Diamond
④ double ellipse ⑤ double diamond.
⑥ Double rect (weak entity)

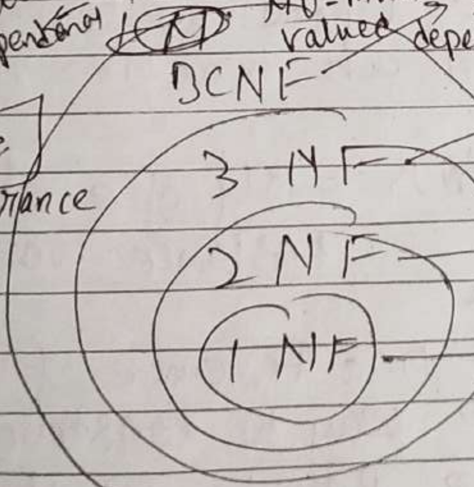
Set Operation

- union (∪): automatically delete duplicat attribute unlike select
- Intersectio (∩): automatically delete duplicat attribut
- except (−): eliminat duplicat attribut befor performing set dif

→ Null values: present arithmetic, set, compost operation.
→ Schema → Desc of data base
→ Instance → collection of data.
→ FD → refer to relnship present b/w attribute.
→ trival FD → $x \to y$: $y \subseteq x$
→ Non trival FD → $x \to y$: $y \not\subseteq x$.

No multi value dep dependency: 4NF → In BCNF, No-multi valued dependency.
BCNF → In 3NF & $x \to y$, x must be CK & SK

Date Referrance



3NF → In 2NF & No transi-tion dependency exist
2NF → In 1NF & NKA are FD on PK
1NF → contain Atomic value

Sequential file organizatio.
Dependency, Sparse
Improv query performanc ② optimi) date costn ③ date acces

Hashing → cluster, primary Index, second Index
multivalued

* Super key :- combination of possible key used to identifies unique tuple
* candidate key :- minimal super key & a super key with no redundant attributes
* Primary key :- One of the candidate key chose by the database designer to identify unique tuple.
* composite key :- combination one & more attribute to identifies unique tuple.

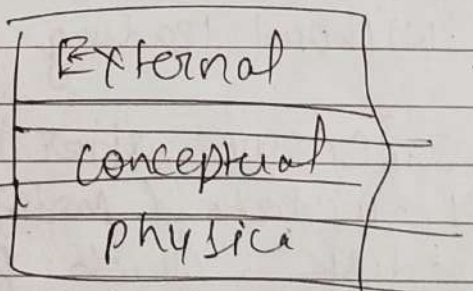
* Max^m Candidate Key = $2^N - 1$ R(AB(n) = 15 [16-1]
Max^m ~~Candidate Key~~ Super = 2 x 4 = 8.

* Adv / feat
(I) Data Sharing
(II) Data security
(III) Data Integrity.
(IV) Data consistency
(V) secure access / concurrent access

* Data Language :-
(I) DDL (II) DCL (III) DML (IV) TCL.

* Types of DBMS :-
(1) RDBMS (MySQL) (2) OODBMS (db4o) (3) NO-SQL DBMS (Mongo DB)

* DBMS - 3 tier Architecture.



(4) (GP-SQL / SQL)

* Data model — conceptual
Representational
Physical

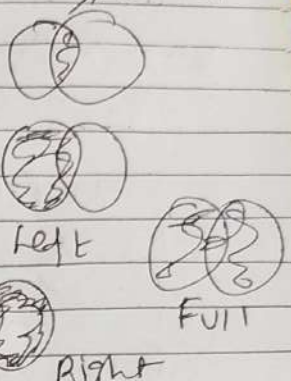
* Height of tree = $h \to 1$
* node = max^m node of l is 2^l.
* node = max^m no of node h is $2^h - 1$.
* node = min^m no of node is h.
* $n = e + 1$
* Height of complete binary. $\log n + 1$

Data Base SQL {General purpose query lang}

CRUD MySQL {RDBMS}

Creat, Read, Update, delete.

- Inner join
- Outer join — left join, Right join, Full join
- ~~Left join~~
- BCNF, 3 NF, 2 NF, 1 NF, 4 NF
- DDL, DCL, DML, TCL
- ACID → Durability.
  - Atomicity
  - Isolation
  - Consistency
- Atomity :- entire transaction take place at once or will not, considered as a single unit.



- consistency :- Integrity constraint must be maintain so, that database is consistent befor & after transaction. No partial transaction.
- Isolation :- Multiple transaction take place con-currently without leading to inconsistency in database.

* Durability :- ensure after transaction is executed the update & modifide data base is written on disk, which persist even if system failure occurs.
* DQL → data query language select. w, nL.
* DDL → Create, delete, Modify date base.
* DML → insert, update, delete tuples
* DCL → Consist state use to control access. (grant & revoke)
* TCL → control transaction to maintain consi

* Hash Table [h(k) = k mod m] ✓
O(1) → Insert / delete / search.

* Tree :- [Non-linear Data Structure]
→ Binary, Full, complete, perfect
✓ (0 or 2) last node complet
at most two every node left node
children except leave filled, All level completely filled
(left & right) has two child (left to right) & node of last level as possible

AVL, B-Tree, BST left
height b/w (inorder traversal) even left node
subtree data store) less than
atmost root, right
one greater than
root

B+ tree

~~Storing data pointers~~
used for indexing by storing data pointer only at the leaf nodes.

I/P :-
Adv Insertion / deletion / traversal / searching.
→ Hierchical representation of data
→ Fast searching & traversal.

Disadv → Memory overhead due to
→ storing pointer or reference
→ Not efficient for storing large amt.

* Best/Avg BST → Array
→ Inorder traversal.
* O(logn) → Search, Insert, dele
Worst O(n) → .

* Minimum Spanning :- Subgraph of a connected, undirected graph, include vertices & edge.

* Use :- XML or JSON, Decision tree, Clustering, File System.

* Stack overflow :
Overflow , top = stack.size-1
underflow top = -1.

* queue :- Queue :- [BFS, FIFO]
* simple — linear — add to rear & retrieve from front
* circular — End queue is connected to front
* priority — element are retrieve as per priority
* Double-ended — Insertion/delet perform from both end

APP :
* CPU scheduling, Disk scheduling

Enqueue(I) & dequeue(I), peek()

overflow — Front = rear+1
underflow — Front = rear-1
(max size)

* Dis adv → limited access to ~~[illegible]~~ element
* not use for non-sequential order
* element not ~~[illegible]~~ retrier randomly.

* Heap :
→ complete binary tree.
→ each node ≥ child node.
→ Max-heap → root node max^n
→ min-heap → root node min
O(log n) → inser/dele. O(1) → for max^n & min^n
All

APP. :
* Priority Queues * Sorting * Huffman Coding
* Dijkstra's algorithm * Networking route

* BST → heap [In-order travel]

Adv :-
* Efficient insertion & extraction O(log n)
* sorting O(n log n)
* Dijkstra's algo.

Tech Interview :
(DSA)

* LL [Linked List] → DMA, Stack & queue
  garbage collector.

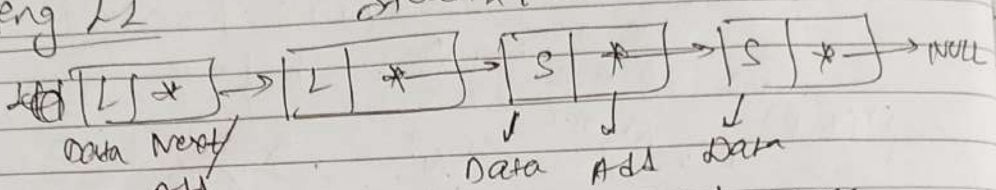
* Disadvantage
  slow random access, more memory, difficult to bug, Not cache-friendly.

* Array Adv → value retrieval O(1), initialize
  Dis → deletion, more memory ~~value~~ on node.

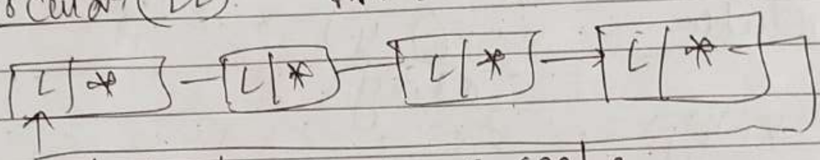
① monitoring & control system
Stack & queue.

② matrix

Sing LL

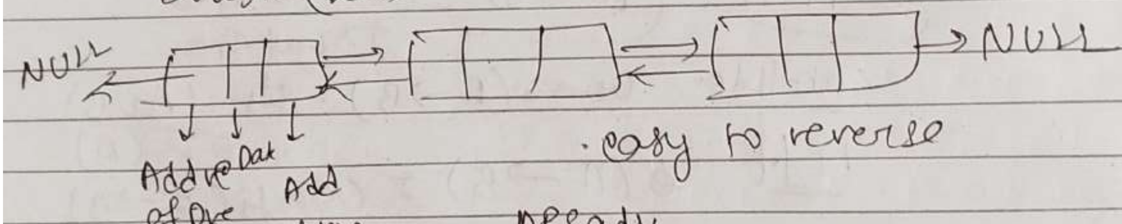


Circular (LL) APP: → OS, web browsers



Double (LL) APP:- OS, cache
traverse from both direction
F → L & L → F

Double (LL)



· easy to reverse

Dis adv
① Required more spaces
② More time insertion & deletion

* Stack : [DFS, LIFO]
  push pop peek O(1)

APP :
① Infix to post fix conversion
② Simplify algebric expression
③ parsing.
④ Balance Parenthesis expression.