DSA by Shradha Didi & Aman Bhaiya

Meet us on Youtube (Apna College)

Easy	Ideal Time : 5-10 mins	
Medium	Ideal Time : 15-20 mins	5 Questions each D
Hard	Ideal Time : 40-60 mins (based on Qs) | 88 Qs	

Topics	Question (375)	Companies
Arrays	Maximum and Minimum Element in an Array	ABCO Accolite Amazon Cisco Hike Microsoft Sna
Arrays	Reverse the Array	Infosys Moonfrog Labs
Arrays	Maximum-Subarray	Microsoft + Facebook Interview Qs
Arrays	Contains Duplicate	Amazon Interview Qs
Arrays	Chocolate Distribution Problem	Amazon Interview Qs
Arrays	Search in Rotated Sorted Array	Microsoft Google Adobe Amazon D-E-Shaw Flip
Arrays	Next Permutation	Uber + Goldman Sachs + Adobe Interview Qs
Arrays	Best time to Buy and Sell Stock	Amazon D-E-Shaw Directi Flipkart Goldman Sach
Arrays	Repeat and Missing Number Array	Amazon Interview Qs
Arrays	Kth-Largest Element in an Array	Amazon Microsoft Walmart Adobe
Arrays	Trapping Rain Water	Samsung Interview Qs
Arrays	Product of Array Except Self	Microsoft + Facebook Interview Qs
Arrays	Maximum Product Subarray	Amazon D-E-Shaw Microsoft Morgan Stanley OY
Arrays	Find Minimum in Rotated Sorted Array	Adobe Amazon Microsoft Morgan Stanley Sams
Arrays	Find Pair with Sum in Sorted & Rotated Array	Microsoft + Google + Apple Interview Qs
Arrays	3Sum	Adobe Amazon Microsoft Morgan Stanley Sams
Arrays	Container With Most Water	Flipkart + Dunzo Interview Qs
Arrays	Given Sum Pair	Infosys + Amazon + Flipkart Interview Qs
Arrays	Kth - Smallest Element	ABCO Accolite Amazon Cisco Hike Microsoft Sna
Arrays	Merge Overlapping Intervals	Google Interview Qs
Arrays	Find Minimum Number of Merge Operations to Make an Array Palindrome	Amazon
Arrays	Given an Array of Numbers Arrange the Numbers to Form the Biggest Number	Barclays Interview Qs
Arrays	Space Optimization Using Bit Manipulations	Amazon
Arrays	Subarray Sum Divisible K	Snapdeal Microsoft
Arrays	Print all Possible Combinations of r Elements in a Given Array of Size n	Amazon
Arrays	Mo's Algorithm	Microsoft
Strings	Valid Palindrome	Amazon Cisco D-E-Shaw Facebook FactSet Morg
Strings	Valid Anagram	Nagarro Media.net Directi Google Adobe Flipka
Strings	Valid parentheses	Google Interview Qs
Strings	Remove Consecutive Characters	Samsung + Adobe
Strings	Longest Common Prefix	Adobe + Grofers + Dunzo Interview Qs
Strings	Convert a Sentence into its Equivalent Mobile Numeric Keypad Sequence	Adobe
Strings	Print all the Duplicates in the Input String	Ola + Amdocs IQ
Strings	Longest Substring without Repeating Characters	Morgan Stanley + Amazon IQ
Strings	Longest Repeating Character Replacement	Amazon Google
Strings	Group Anagrams	Samsung + Adobe + Amazon Interview Qs
Strings	Longest Palindromic Substring	Microsoft + Google + Samsung + Visa IQ
Strings	Palindromic Substrings	Microsoft IQ
Strings	Next Permutation	Adobe + Goldman Sachs + Uber
Strings	Count Palindromic Subsequences	Myntra Interview Qs
Strings	Smallest Window in a String Containing all the Characters of Another String	Microsoft + Amazon IQ
Strings	Wildcard String Matching	Microsoft + Amazon + Ola IQ
Strings	Longest Prefix Suffix	Flipkart + Swiggy IQ
Strings	Rabin-Karp Algorithm for Pattern Searching	Microsoft
Strings	Transform One String to Another using Minimum Number of Given Operation	Directi
Strings	Minimum Window Substring	Amazon Google MakeMyTrip Streamoid Techno
Strings	Boyer Moore Algorithm for Pattern Searching	Amdocs
Strings	Word Wrap	Microsoft
2D Arrays	Zigzag (or diagonal) Traversal of Matrix	Amazon
2D Arrays	Set Matrix Zeroes	Amazon Microsoft
2D Arrays	Spiral Matrix	Flipkart + Apple + Societe Generale IQ
2D Arrays	Rotate Image	Microsoft Paytm Samsung Adobe
2D Arrays	Word Search	Google + Ola + Goldman Sachs IQ
2D Arrays	Find the Number of Islands | Set 1 (Using DFS)	Microsoft + Uber + Apple + Amazon IQ
2D Arrays	Given a Matrix of 'O' and 'X', Replace 'O' with 'X' if Surrounded by 'X'	Google
2D Arrays	Find a Common Element in all Rows of a Given Row-Wise Sorted Matrix	MAQ Software Microsoft VMWare
2D Arrays	Create a Matrix with Alternating Rectangles of O and X	MAQ VMWare
2D Arrays	Maximum Size Rectangle of all 1s	Amazon Microsoft

Backtracking	Backtracking Set 2 Rat in a Maze	Microsoft Amazon
Backtracking	Combinational Sum	Adobe Amazon Microsoft
Backtracking	Crossword-Puzzle	Microsoft
Backtracking	Longest Possible Route in a Matrix with Hurdles	Microsoft
Backtracking	Printing all solutions in N-Queen Problem	Accolite Amazon Amdocs D-E-Shaw MAQ Softwa
Backtracking	Solve the Sudoku	Amazon Directi Flipkart MakeMyTrip MAQ Softw
Backtracking	Partition Equal Subset Sum	Amazon + Adobe + Accolite + Traveloka
Backtracking	M Coloring Problem	Amazon
Backtracking	Knight Tour	IBM
Backtracking	Soduko	Amazon + Adobe + Accolite + Traveloka
Backtracking	Remove Invalid Parentheses	Uber
Backtracking	Word Break Problem using Backtracking	
Backtracking	Print all Palindromic Partitions of a String	Facebook Amazon Microsoft
Backtracking	Find Shortest Safe Route in a Path with Landmines	Facebook Amazon Microsoft
Backtracking	Partition of Set into K Subsets with Equal Sum	Amazon
Backtracking	Backtracking set-7 hamiltonian cycle	Amazon
Backtracking	tug-of-war	Google
Backtracking	Maximum Possible Number by doing at most K swaps	Amazon + Adobe + Accolite + Traveloka
Backtracking	Backtracking set-8 solving cryptarithmetic puzzles	Goldman Sachs
Backtracking	Find paths from corner cell to middle cell in maze	Meta
Backtracking	Arithmetic Expressions	Flipkart
Linked List	Reverse Linked List	Sprinklr
Linked List	Linked List Cycle	Accolite Amazon D-E-Shaw Hike Lybrate Mahind
Linked List	Merge Two Sorted Lists	Accolite Amazon Belzabar Brocade FactSet Flipk
Linked List	Delete without Head node	Amazon Goldman Sachs Kritikal Solutions Micro
Linked List	Remove duplicates from an unsorted linked list	Amazon Intuit
Linked List	Sort a linked list of 0s-1s-or-2s	Microsoft Amazon MakeMyTrip
Linked List	Multiply two numbers represented linked lists	Amazon
Linked List	Remove nth node from end of list	Accolite Adobe Amazon Citicorp Epic Systems F
Linked List	Reorder List	Amazon Microsoft OYO Rooms Intuit
Linked List	Detect and remove loop in a linked list	Accolite Amazon D-E-Shaw Hike Lybrate Mahind
Linked List	Write a Function to get the Intersection Point of two Linked Lists	Amazon
Linked List	Flatten a linked list with next and child pointers	Google
Linked List	Linked list in zig-zag fashion	Micorsoft
Linked List	Reverse a doubly linked list	Walmart
Linked List	Delete nodes which have a greater value on right side	Amazon
Linked List	Segregate even and odd Elements in a Linked List	Walmart
Linked List	Point to next higher value node in a linked list with an Arbitrary Pointer	GeekyAnts
Linked List	Rearrange a given linked list in place	Ola Uber
Linked List	Sort Biotonic Doubly Linked Lists	Morgan Stanley
Linked List	Merge K Sorted Lists	Microsoft+ Ola+ eBay
Linked List	Merge sort for linked list	Accolite Adobe Amazon MAQ Software Microso
Linked List	Quicksort on singly-linked list	Paytm
Linked List	Sum of two linked lists	Accolite Amazon Flipkart MakeMyTrip Microsoft
Linked List	Flattening a linked list	24*7 Innovation Labs Amazon Drishti-Soft Flipk
Linked List	Clone a linked list with next and random Pointer	Triology
Linked List	Subtract two numbers represented as linked lists	Amazon Goldman Sachs
Stacks & Queues	Implement two stacks in an Array	24*7 Innovation Labs Microsoft Samsung Snapd
Stacks & Queues	Evaluation of Postfix Expression	Amazon + Google + Facebook
Stacks & Queues	Implement Stack using Queues	Facebook
Stacks & Queues	Queue Reversal	Amazon + Morgain Stanley
Stacks & Queues	Implement Stack Queue using Deque	Microsoft +Atlassian
Stacks & Queues	Reverse first k elements of queue	Microsoft + Amdocs
Stacks & Queues	Design Stack with Middle Operation	MaQ Software
Stacks & Queues	Infix to Postfix	Amazon + Samsung + Paytm + Vmware inc
Stacks & Queues	Design and Implement Special stack	Amazon Google Microsoft Visa Goldman Sachs
Stacks & Queues	Longest Valid String	Google Microsoft
Stacks & Queues	Find if an expression has duplicate parenthesis or not	Flipkart Oracle OYO Rooms Snapdeal Walmart Y
Stacks & Queues	Stack permutations check if an array is stack permutation of other	Visa
Stacks & Queues	Count natural numbers whose permutation greater number	Amazon
Stacks & Queues	Sort a stack using Recursion	Amazon Goldman Sachs IBM Intuit Kuliza Yahoo
Stacks & Queues	Queue based approach for first non repeating character in a stream	Microsoft Flipkart
Stacks & Queues	The Celebrity Problem	Google + Visa + Apple
Stacks & Queues	Next larger Element	Visa
Stacks & Queues	Distance of nearest cell	Flipkar + Facebook
Stacks & Queues	Rotten-oranges	Facebook
Stacks & Queues	Next smaller element	Codenation
Stacks & Queues	Circular-tour	Codenation Flipkart
Stacks & Queues	Efficiently implement k stacks single array	Flipkart

Greedy	Minimum edges to reverse to make path from a source to a destination	
Greedy	Minimize Cash Flow among a given set of friends who have borrowed money from each other	
Greedy	Minimum Cost to cut a board into squares	Maccafe
Binary Trees	Maximum Depth of Binary Tree	Amazon Cadence India CouponDunia D-E-Shaw
Binary Trees	Reverse Level Order Traversal	Amazon + Microsoft + flipkart + Adobe
Binary Trees	Subtree of Another Tree	Amazon + Microsoft + Facebook
Binary Trees	Invert Binary Tree	Amazon Hike
Binary Trees	Binary Tree Level Order Traversal	Accolite Adobe Amazon Cisco D-E-Shaw Flipkart
Binary Trees	Left View of Binary Tree	Microsoft + Adobe + Cisco Networking Academy
Binary Trees	Right View of Binary Tree	Amdocs
Binary Trees	ZigZag Tree Traversal	Amazon Cisco FactSet Hike Snapdeal Walmart
Binary Trees	Create a mirror tree from the given binary tree	Accolite Adobe Amazon Belzabar EBay Goldman
Binary Trees	Leaf at same level	Amazon
Binary Trees	Check for Balanced Tree	Amazon Walmart Microsoft
Binary Trees	Transform to Sum Tree	Amazon FactSet Microsoft Samsung Walmart
Binary Trees	Check if Tree is Isomorphic	Amazon Microsoft
Binary Trees	Same Tree	Amazon Microsoft Flipkart
Binary Trees	Construct Binary Tree from Preorder and Inorder Traversal	Accolite Amazon Microsoft
Binary Trees	Height of Binary Tree	Amazon Cadence India CouponDunia D-E-Shaw
Binary Trees	Diameter of a Binary Tree	Amazon Microsoft OYO Rooms
Binary Trees	Top View of Binary Tree	Microsoft + Adobe + Expedia Group
Binary Trees	Bottom View of Binary Tree	DE Shaw India
Binary Trees	Diagonal Traversal of Binary Tree	Amazon Microsoft
Binary Trees	Boundary Traversal of binary tree	Accolite Amazon FactSet Hike Kritikal Solutions
Binary Trees	Construct Binary Tree from String with Brackets	Microsoft Morgan Stanley OYO Rooms Payu Sa
Binary Trees	Minimum swap required to convert binary tree to binary search tree	Adobe Amazon
Binary Trees	Duplicate subtree in Binary Tree	Google
Binary Trees	Check if a given graph is tree or not	Microsoft Amazon
Binary Trees	Lowest Common Ancestor in a Binary Tree	Accolite Amazon American Express Cisco Exped
Binary Trees	Min distance between two given nodes of a Binary Tree	Amazon Linkedin MakeMyTrip Ola Cabs Qualco
Binary Trees	Duplicate Subtrees	Ola
Binary Trees	Kth ancestor of a node in binary tree	Josh Technology Group
Binary Trees	Binary Tree Maximum Path Sum	Samsung + Facebook
Binary Trees	Serialize and Deserialize Binary Tree	Flipkart InMobi Linkedin MAQ Software Microso
Binary Trees	Binary Tree to DLL	Accolite Amazon Goldman Sachs Microsoft Mor
Binary Trees	Print all k-sum paths in a binary tree	Accolite Amazon Goldman Sachs
Binary Search Trees	Lowest Common Ancestor of a Binary Search Tree	Accolite Amazon Flipkart MAQ Software Microso
Binary Search Trees	Binary Search Tree | Set 1 (Search and Insertion)	Accolite Amazon Microsoft Paytm Samsung
Binary Search Trees	Minimum element in BST	Microsoft
Binary Search Trees	Predecessor and Successor	Google + Adobe + Goladman Sachs + Direct
Binary Search Trees	Check whether BST contains Dead End	Walmart
Binary Search Trees	Binary Tree to BST	HSBC
Binary Search Trees	Kth largest element in BST	Accolite Amazon Samsung SAP Labs Microsoft
Binary Search Trees	Validate Binary Search Tree	OYO Rooms Qualcomm Samsung Snapdeal VM
Binary Search Trees	Kth Smallest Element in a BST	Accolite Amazon Google
Binary Search Trees	Delete Node in a BST	Adobe Barclays
Binary Search Trees	Flatten BST to sorted list	Microsoft
Binary Search Trees	Preorder to Postorder	Amazon Linkedin Flipkart
Binary Search Trees	Count BST nodes that lie in a given range	D-E-Shaw Google
Binary Search Trees	Populate Inorder Successor for all Nodes	Sap labs
Binary Search Trees	Convert Normal BST to Balanced BST	Paytm
Binary Search Trees	Merge two BSTs	DE Shaw India
Binary Search Trees	Given n appointments, find all conflicting appointments	Samsung
Binary Search Trees	Replace every element	Samsung
Binary Search Trees	Construct BST from given preorder traversal	Adobe Morgan Stanley Microsoft
Binary Search Trees	Find median of BST in O(n) time and O(1) space	Amazon
Binary Search Trees	Largest BST in a Binary Tree	Amazon D-E-Shaw Samsung Microsoft Flipkart
Heaps & Hashing	Choose k array elements such that difference of maximum and minimum is minimized	
Heaps & Hashing	Heap Sort	Adobe
Heaps & Hashing	Top K Frequent Elements	Amazon Microsoft
Heaps & Hashing	k largest elements in an array	Amazon Microsoft Walmart Adobe
Heaps & Hashing	Next Greater Element	Amazon + Microsoft + Flipkart + Adobe
Heaps & Hashing	K'th Smallest/Largest Element in Unsorted Array	ABCO Accolite Amazon Cisco Hike Microsoft Sn
Heaps & Hashing	Find the maximum repeating number in O(n) time and O(1) extra space	Accolite Amazon
Heaps & Hashing	K-th smallest element after removing some integers from natural numbers	ABCO Accolite Amazon Cisco Hike Microsoft Sn
Heaps & Hashing	Find k closest elements to a given value	Amazon OYO Rooms
Heaps & Hashing	K'th largest element in a stream	Amazon Cisco Hike OYO Rooms Walmart Micro

Graphs	Steps by Knight	Samsung
Graphs	Clone graph	Google + MAQ Software + Apple + Facebook
Graphs	Number of Operations to Make Network Connected	Samsung
Graphs	Dijkstra's shortest path algorithm	Amazon
Graphs	Topological Sort	Amazon + Google + Flipkart + Oyo + Fipkart + Sa
Graphs	Oliver and the Game	Sharechat + Directi
Graphs	Minimum time taken by each job to be completed given by a Directed Acyclic Graph	Amazon
Graphs	Find whether it is possible to finish all tasks or not from given dependencies	Directi + Sharechat
Graphs	Find the number of islands	Razorpay
Graphs	Prim's Algo	Visa
Graphs	Negative Weighted Cycle	Amazon
Graphs	Floyd Warshall	Google + Uber
Graphs	Graph Coloring	Morgan Stanley
Graphs	Snakes and Ladders	Goldman Sachs +Makemytrip
Graphs	Kosaraju's Theorem	Paytm
Graphs	Journey to moon	Lenksart + Payload
Graphs	Vertex Cover	Intuit
Graphs	M Coloring Problem	Uber
Graphs	Cheapest Flights Within K Stops	Uber + Paypal
Graphs	Find if there is a path of more than k length from a source	Cisco + Intuit
Graphs	Bellman Ford	Sharechat + Directi
Graphs	Bipartitie Graph	Microsoft Flipkart
Graphs	Word-Ladder	Microsoft
Graphs	Allen Dictionary	Samsung
Graphs	Kruskals MST	Amazon Cisco Samsung
Graphs	Total number spanning trees graph	Amazon Cisco Samsung Microsoft Flipkart
Graphs	Travelling Salesman	Google + Microsoft + Opera
Graphs	Find longest path directed acyclic graph	Google
Graphs	Two Clique Problem	Microsoft
Graphs	Minimise the cash flow	Intuit + Uber
Graphs	Chinese postman	Intuit
Graphs	Water Jug	Intuit + Uber
Graphs	Water Jug 2	MakeMyTrip MAQ Software
Tries	Construct a trie from scratch	Accolite Amazon D-E-Shaw FactSet Microsoft
Tries	Print unique rows in a given boolean matrix	Amazon Zoho
Tries	Word Break Problem | (Trie solution)	Amazon Google Hike IBM MAQ Software Micros
Tries	Given a sequence of words, print all anagrams together	Amazon D-E-Shaw Goldman Sachs Morgan Stan
Tries	Find shortest unique prefix for every word in a given list	Microsoft Google
Tries	Implement a Phone Directory	Amazon + Microsoft + Snapdeal
DP	Knapsack with Duplicate Items	Amazon
DP	BBT counter	Microsoft
DP	Reach a given score	Samsung
DP	Maximum difference of zeros and ones in binary string	Ola
DP	Climbing Stairs	Intuit
DP	Permutation Coefficient	Amazon
DP	Longest Repeating Subsequence	Google + Amazon
DP	Pairs with specific difference	Ola
DP	Longest subsequence-1	Amazon
DP	Coin Change	Microsoft+ Samsung + Barclays + Apple + Adobe
DP	LIS	Amazon + Google + Facebook + Fidelity Internati
DP	Longest Common Subsequence	Siemens + Amazon + Google
DP	Word Break	Amazon + Google + Microsoft + Walmart + Apple
DP	Combination Sum IV	Adobe Amazon Microsoft
DP	House Robber	Apple + Uber
DP	Houe Robber 2	Arrays Dynamic Programming
DP	Decode Ways	Adobe + Uber
DP	Unique Paths	Google + Microsoft
DP	Jumps Game	Facebook Amazon Microsoft Google
DP	Knapsack Problem	Amazon Directi Flipkart GreyOrange Microsoft M
DP	nCr	Google
DP	Catalan Number	Amazon + Google
DP	Edit Distance	Google + Goldman Sachs + Citrix
DP	Subset Sum	Amazon + Google
DP	Gold mine	Samsung
DP	Assembly Line Scheduling	Goldman Sachs
DP	Maximize The Cut Segments	Amazon OYO Rooms Microsoft
DP	Maximum sum increasing subsequence	Amazon Morgan Stanley Microsoft
DP	Count all subsequences having product less than K	Goldman Sachs
DP	Maximum sum increasing subsequence	Amazon Morgan Stanley Microsoft

Bit Manipulation	Find the two non-repeating elements in an array of repeating elements	Accolite Amazon FactSet Google MakeMyTrip Mi
Bit Manipulation	Program to find whether a no is power of two	Adobe
Bit Manipulation	Find position of the only set bit	Microsoft
Bit Manipulation	Count number of bits to be flipped to convert A to B	Maq Software
Bit Manipulation	Count total set bits in all numbers from 1 to n	Microsoft
Bit Manipulation	Copy set bits in a range	Facebook
Bit Manipulation	Calculate square of a number without using *, / and pow()	Amazon
Bit Manipulation	Divide two integers without using multiplication, division and mod operator	Microsoft
Bit Manipulation	Power Set	Google + Adobe + Paytm
Segment Trees	Range Sum Query - Immutable	
Segment Trees	Range Minimum Query	Google Interview Qs
Segment Trees	Range Sum Query - Mutable	Alibaba
Segment Trees	Create Sorted Array through Instructions	Samsung + Accolite
Segment Trees	Count of Range Sum	Walmart
Segment Trees	Count of Smaller Numbers After Self	Codenation Google

Java - Introduction to Programming
Lecture 1

Installation & First Program

1. Install Java

a. Install JDK (https://www.oracle.com/in/java/technologies/javase-downloads.html)
b. Install IntelliJ (https://www.jetbrains.com/idea/download/#section=mac)
OR
b. Install Visual Studio Code (VS Code) - Prefer THIS (https://code.visualstudio.com/download)

2. Sample Code

Functions

A function is a block of code which takes some input, performs some operations and returns some output.
The functions stored inside classes are called methods.
The function we have used is called main.

Class

A class is a group of objects which have common properties. A class can have some properties and functions (called methods).
The class we have used is Main.

3. Our 1st Program

```
public class Main {

   public static void main(String[] args) {
  // Our 1st Program
       System.out.println("Hello World");
   }
}
```

Apna College

Java - Introduction to Programming
Lecture 2

Variables & Data Types

1. Variables

A variable is a container (storage area) used to hold data.
Each variable should be given a unique name (identifier).

```
package com.apnacollege;

public class Main {

    public static void main(String[] args) {
        // Variables
         String name = "Aman";
         int age = 30;

         String neighbour = "Akku";
         String friend = neighbour;
    }
}
```

2. Data Types

Data types are declarations for variables. This determines the type and size of data associated with variables which is essential to know since different data types occupy different sizes of memory.

There are 2 types of Data Types :

- Primitive Data types : to store simple values
- Non-Primitive Data types : to store complex values

Primitive Data Types

These are the data types of fixed size.

Apna College

Data Type	Meaning	Size (in Bytes)	Range
byte	2's complement integer	1	-128 to 127
short	2's complement integer	2	-32K to 32K
int	Integer numbers	4	-2B to 2B
long	2's complement integer (larger values)	8	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	Floating-point	4	Upto 7 decimal digits
double	Double Floating-point	8	Upto 16 decimal digits
char	Character	2	a, b, c .. A, B, C .. @, #, $ ..
bool	Boolean	1	True, false

Non-Primitive Data Types
These are of variable size & are usually declared with a 'new' keyword.

Eg : String, Arrays

```
String name = new String("Aman");
int[] marks = new int[3];
marks[0] = 97;
marks[1] = 98;
marks[2] = 95;
```

3. Constants

A constant is a variable in Java which has a fixed value i.e. it cannot be assigned a different value once assigned.

Apna College

```java
package com.apnacollege;

public class Main {

    public static void main(String[] args) {
  // Constants
        final float PI = 3.14F;
    }
}
```

Homework Problems

1. Try to declare meaningful variables of each type. Eg – a variable named age should be a numeric type (int or float) not byte.
2. Make a program that takes the radius of a circle as input, calculates its radius and area and prints it as output to the user.
3. Make a program that prints the table of a number that is input by the user.

   (HINT – You will have to write 10 lines for this but as we proceed in the course you will be studying about 'LOOPS' that will simplify your work A LOT!)

KEEP LEARNING & KEEP PRACTICING :)

Apna College

Java - Introduction to Programming

Lecture 3

1. Conditional Statements 'if-else'

The if block is used to specify the code to be executed if the condition specified in if is true, the else block is executed otherwise.

```
int age = 30;
if(age > 18) {
   System.out.println("This is an adult");
} else {
   System.out.println("This is not an adult");
}
```

2. Conditional Statements 'switch'

Switch case statements are a substitute for long if statements that compare a variable to multiple values. After a match is found, it executes the corresponding code of that value case.

The following example is to print days of the week:

```
int n = 1;
switch(n) {
   case 1 :
       System.out.println("Monday");
       break;
   case 2 :
       System.out.println("Tuesday");
       break;
   case 3 :
       System.out.println("Wednesday");
       break;
   case 4 :
       System.out.println("Thursday");
       break;
   case 5:
       System.out.println("Friday");
       break;
   case 6 :
       System.out.println("Saturday");
       break;
   default :
       System.out.println("Sunday");
}
```

Apna College

Homework Problems

1. Make a Calculator. Take 2 numbers (a & b) from the user and an operation as follows :

   1 : + (Addition) a + b

   - 2 : - (Subtraction) a - b
   - 3 : * (Multiplication) a * b
   - 4 : / (Division) a / b
   - 5 : % (Modulo or remainder) a % b

   Calculate the result according to the operation given and display it to the user.

2. Ask the user to enter the number of the month & print the name of the month. For eg - For '1' print 'January', '2' print 'February' & so on.

KEEP LEARNING & KEEP PRACTICING :)

Apna College

Java - Introduction to Programming
Lecture 4

Loops

A loop is used for executing a block of statements repeatedly until a particular condition is satisfied. A loop consists of an initialization statement, a test condition and an increment statement.

For Loop

The syntax of the for loop is :

```
for (initialization; condition; update) {
 // body of-loop
}
```

```
for (int i=1; i<=20; i++) {
   System.out.println(i);
}
```

While Loop

The syntax for while loop is :

```
while(condition) {
 // body of the loop
}
```

```
int i = 0;
while(i<=20) {
   System.out.println(i);
   i++;
}
```

Do-While Loop

The syntax for the do-while loop is :

```
do {
 // body of loop;
}
while (condition);
```

```
int i = 0;
do {
   System.out.println(i);
```

Apna College

```
    i++;
} while(i<=20);
```

Homework Problems

1. Print all even numbers till n.
2. Run

```
for(; ;) {

            System.out.println("Apna College");

        }
```

loop on your system and analyze what happens. Try to think of the reason for the output produced.

3. Make a menu driven program. The user can enter 2 numbers, either 1 or 0.

   If the user enters 1 then keep taking input from the user for a student's marks(out of 100).

   If they enter 0 then stop.

   If he/ she scores :

   Marks >=90 -> print "This is Good"

   89 >= Marks >= 60 -> print "This is also Good"

   59 >= Marks >= 0 -> print "This is Good as well"

   Because marks don't matter but our effort does.

   (Hint : use do-while loop but think & understand why)

BONUS

Qs. Print if a number is prime or not (Input n from the user).

[In this problem you will learn how to check if a number is prime or not]

Apna College

Homework Solution (Lecture 3)

```
import java.util.*;

public class Conditions {
   public static void main(String args[]) {
       Scanner sc = new Scanner(System.in);
       int a = sc.nextInt();
       int b = sc.nextInt();
       int operator = sc.nextInt();

       /**
        * 1 -> +
        * 2 -> -
        * 3 -> *
        * 4 -> /
        * 5 -> %
        */

       switch(operator) {
           case 1 : System.out.println(a+b);
           break;
           case 2 : System.out.println(a-b);
           break;
           case 3 : System.out.println(a*b);
           break;
           case 4 : if(b == 0) {
                       System.out.println("Invalid Division");
                   } else {
                       System.out.println(a/b);
                   }
           break;
           case 5 : if(b == 0) {
                       System.out.println("Invalid Division");
                   } else {
                       System.out.println(a%b);
                   }
           break;
           default : System.out.println("Invalid Operator");
       }
   }
```

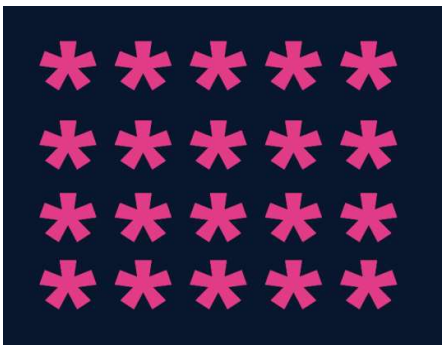
Apna College

```
}
```

Apna College

Java - Introduction to Programming
Lecture 5

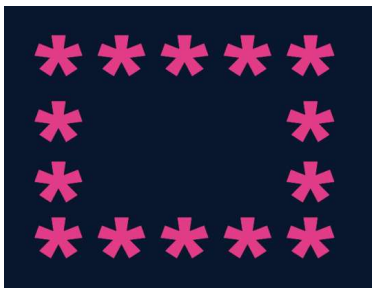
Patterns - Part 1

1. 

```
import java.util.*;

public class Patterns {
   public static void main(String args[]) {
       int n = 5;
       int m = 4;
       for(int i=0; i<n; i++) {
           for(int j=0; j<m; j++) {
               System.out.print("*");
           }
           System.out.println();
       }
   }
}
```

Apna College

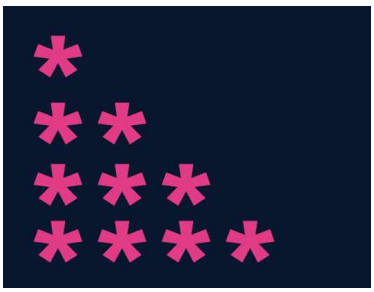
2.



```
import java.util.*;

public class Patterns {
   public static void main(String args[]) {
       int n = 5;
       int m = 4;
       for(int i=0; i<n; i++) {
           for(int j=0; j<m; j++) {
               if(i == 0 || i == n-1 || j == 0 || j == m-1) {
                   System.out.print("*");
               } else {
                   System.out.print(" ");
               }
           }
           System.out.println();
       }
   }
}
```

Apna College

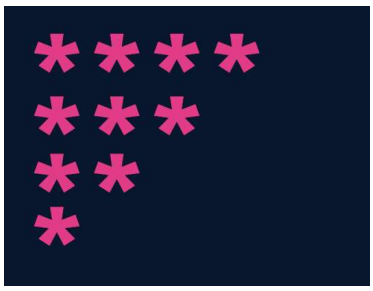
3.



```
import java.util.*;

public class Patterns {
   public static void main(String args[]) {
       int n = 4;

       for(int i=1; i<=n; i++) {
           for(int j=1; j<=i; j++) {
                   System.out.print("*");
           }
           System.out.println();
       }
   }
}
```

Apna College

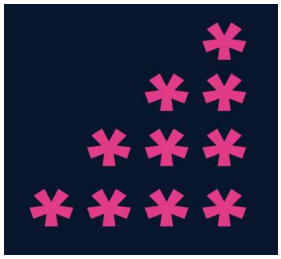
4.



```
import java.util.*;

public class Patterns {
   public static void main(String args[]) {
       int n = 4;

       for(int i=n; i>=1; i--) {
           for(int j=1; j<=i; j++) {
                   System.out.print("*");
           }
           System.out.println();
       }
   }
}
```

Apna College

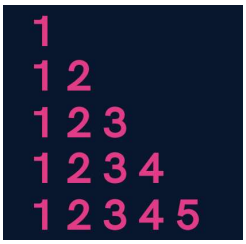
5.



```java
import java.util.*;

public class Patterns {
   public static void main(String args[]) {
       int n = 4;

       for(int i=n; i>=1; i--) {
           for(int j=1; j<i; j++) {
               System.out.print(" ");
           }

           for(int j=0; j<=n-i; j++) {
               System.out.print("*");
           }
           System.out.println();
       }
   }
}
```

Apna College

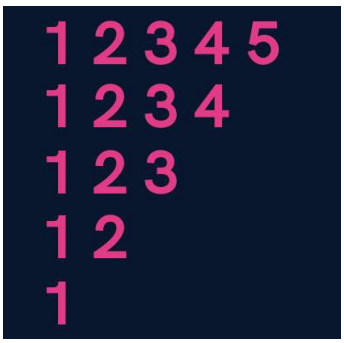
6.




```java
import java.util.*;

public class Patterns {
   public static void main(String args[]) {
       int n = 5;

       for(int i=1; i<=n; i++) {
           for(int j=1; j<=i; j++) {
               System.out.print(j);
           }
           System.out.println();
       }
   }
}
```

Apna College

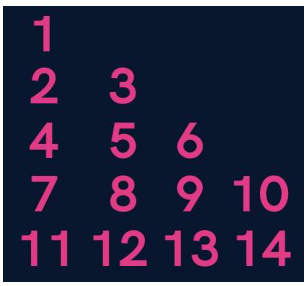
7.




```java
import java.util.*;

public class Patterns {
   public static void main(String args[]) {
       int n = 5;

       for(int i=n; i>=1; i--) {
           for(int j=1; j<=i; j++) {
               System.out.print(j);
           }
           System.out.println();
       }
   }
}
```

Apna College

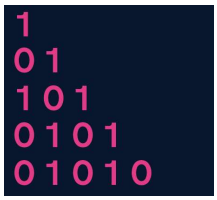
8.




```java
import java.util.*;

public class Patterns {
   public static void main(String args[]) {
       int n = 5;
       int number = 1;

       for(int i=1; i<=n; i++) {
           for(int j=1; j<=i; j++) {
               System.out.print(number+" ");
               number++;
           }
           System.out.println();
       }
   }
}
```

Apna College

9.

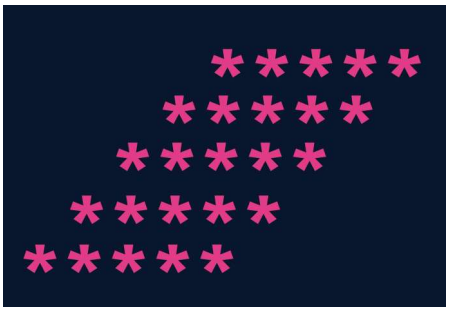



```java
import java.util.*;

public class Patterns {
   public static void main(String args[]) {
       int n = 5;

       for(int i=1; i<=n; i++) {
           for(int j=1; j<=i; j++) {
               if((i+j) % 2 == 0) {
                   System.out.print(1+" ");
               } else {
                   System.out.print(0+" ");
               }
           }
           System.out.println();
       }
   }
}
```

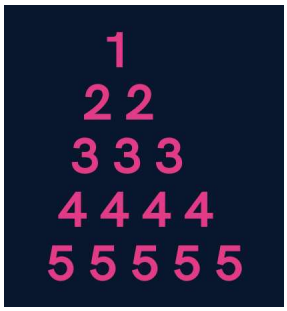
Apna College

Homework Problems (Solutions in next Lecture's Video)

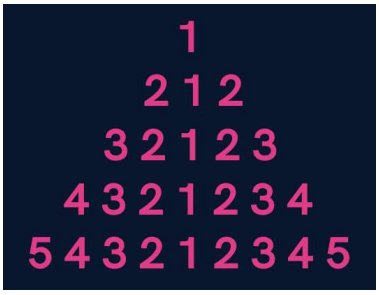
1. Print a solid rhombus.



2. Print a number pyramid.



3. Print a palindromic number pyramid.



Apna College

Homework Solution (Lecture 4)

1. Print all even numbers till n.

```
public class Solutions {
    public static void main(String args[]) {
        int n = 25;

        for(int i=1; i<=n; i++) {
            if(i % 2 == 0) {
                System.out.println(i);
            }
        }
    }
}
```

3. Make a menu driven program. The user can enter 2 numbers, either 1 or 0.
   If the user enters 1 then keep taking input from the user for a student's marks(out of 100).

   If they enter 0 then stop.

   If he/ she scores :

   Marks >=90 -> print "This is Good"

   89 >= Marks >= 60 -> print "This is also Good"

   59 >= Marks >= 0 -> print "This is Good as well"

   Because marks don't matter but our effort does.

Apna College

(Hint : use do-while loop but think & understand why)

```
import java.util.*;

public class Solutions {
   public static void main(String args[]) {
       Scanner sc = new Scanner(System.in);
       int input;

       do {
           int marks = sc.nextInt();
           if(marks >= 90 && marks <= 100) {
               System.out.println("This is Good");
           } else if(marks >= 60 && marks <= 89) {
               System.out.println("This is also Good");
           } else if(marks >= 0 && marks <= 59) {
               System.out.println("This is Good as well");
           } else {
               System.out.println("Invalid");
           }

           System.out.println("Want to continue ? (yes(1) or no(0))");
           input = sc.nextInt();

       } while(input == 1);
   }
}
```

Qs. Print if a number n is prime or not (Input n from the user).

[In this problem you will learn how to check if a number is prime or not]

```
import java.util.*;

public class Solutions {
   public static void main(String args[]) {
       Scanner sc = new Scanner(System.in);
       int n = sc.nextInt();

       boolean isPrime = true;
       for(int i=2; i<=n/2; i++) {
```

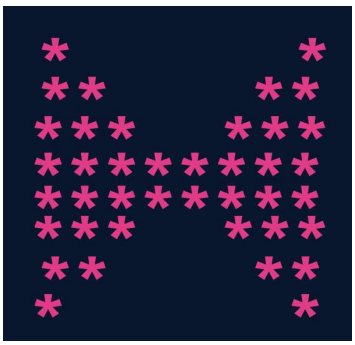
Apna College

```
            if(n % i == 0) {
                isPrime = false;
                break;
            }
        }

        if(isPrime) {
            if(n == 1) {
                System.out.println("This is neither prime not composite");
            } else {
                System.out.println("This is a prime number");
            }
        } else {
            System.out.println("This is not a prime number");
        }
    }
}
```

Apna College

Java - Introduction to Programming
Lecture 6

Patterns - Part 2

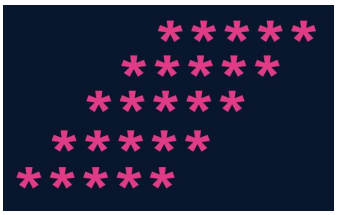
1. 

```
import java.util.*;

public class Solutions {
   public static void main(String args[]) {
       int n = 4;

       //upper part
       for(int i=1; i<=n; i++) {
           for(int j=1; j<=i; j++) {
               System.out.print("*");
           }

           int spaces = 2 * (n-i);
           for(int j=1; j<=spaces; j++) {
               System.out.print(" ");
           }

           for(int j=1; j<=i; j++) {
               System.out.print("*");
           }
           System.out.println();
       }
```

Apna College

```
        //lower part
        for(int i=n; i>=1; i--) {
            for(int j=1; j<=i; j++) {
                System.out.print("*");
            }

            int spaces = 2 * (n-i);
            for(int j=1; j<=spaces; j++) {
                System.out.print(" ");
            }

            for(int j=1; j<=i; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

2.



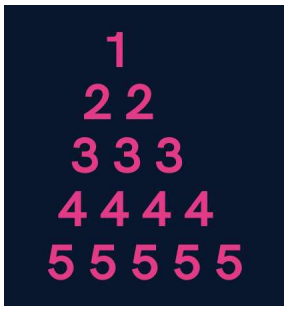
```
import java.util.*;

public class Solutions {
    public static void main(String args[]) {
        int n = 5;
```

Apna College

```
        for(int i=1; i<=n; i++) {
            //spaces
            for(int j=1; j<=n-i; j++) {
                System.out.print(" ");
            }

            //stars
            for(int j=1; j<=n; j++) {
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Apna College

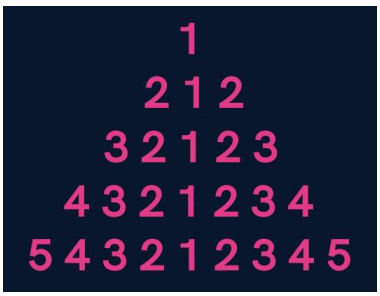
3.




```
import java.util.*;

public class Solutions {
   public static void main(String args[]) {
       int n = 5;

       for(int i=1; i<=n; i++) {
           //spaces
           for(int j=1; j<=n-i; j++) {
               System.out.print(" ");
           }

           //numbers
           for(int j=1; j<=i; j++) {
               System.out.print(i+" ");
           }
           System.out.println();
       }
   }
}
```

Apna College

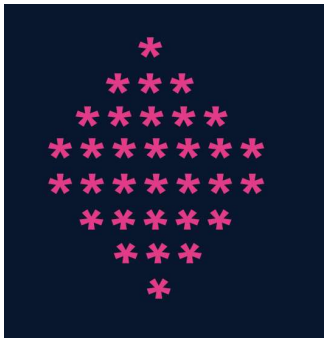
4.




```java
import java.util.*;

public class Solutions {
   public static void main(String args[]) {
       int n = 5;
       for(int i=1; i<=n; i++) {
           //spaces
           for(int j=1; j<=n-i; j++) {
               System.out.print(" ");
           }

           //first part
           for(int j=i; j>=1; j--) {
               System.out.print(j);
           }

           //second part
           for(int j=2; j<=i; j++) {
               System.out.print(j);
           }
           System.out.println();
       }
   }
}
```

Apna College

5.

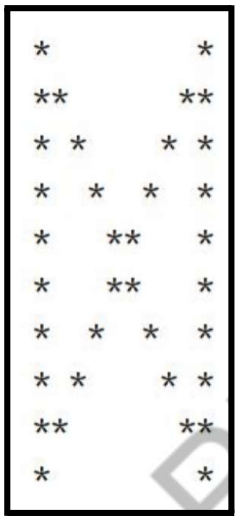


```
import java.util.*;

public class Solutions {
   public static void main(String args[]) {
       int n = 5;

       //upper part
       for(int i=1; i<=n; i++) {
           //spaces
           for(int j=1; j<=n-i; j++) {
               System.out.print(" ");
           }
           for(int j=1; j<=2*i-1; j++) {
               System.out.print("*");
           }
           System.out.println();
       }

       //lower part
       for(int i=n; i>=1; i--) {
           //spaces
           for(int j=1; j<=n-i; j++) {
               System.out.print(" ");
           }
           for(int j=1; j<=2*i-1; j++) {
               System.out.print("*");
           }
           System.out.println();
       }
   }
}
```

Apna College

Homework Problems

1. **Print a hollow Butterfly.**



2. **Print a hollow Rhombus.**

```
  *****
 *     *
*     *
*     *
*****
```

3. **Print Pascal's Triangle.**

```
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

4. **Print half Pyramid.**

```
1
```

Apna College

1 2

1 2 3

1 2 3 4

1 2 3 4 5

5. **Print Inverted Half Pyramid.**

11111

222

33

4

Apna College

Java - Introduction to Programming
Lecture 7

Methods/Functions

A function is a block of code that performs a specific task.
Why are functions used?

a. If some functionality is performed at multiple places in software, then rather than writing the same code, again and again, we create a function and call it everywhere. This helps reduce code redundancy.
b. Functions make maintenance of code easy as we have to change at one place if we make future changes to the functionality.
c. Functions make the code more readable and easy to understand.

The **syntax** for function declaration is :

```
return-type function_name (parameter 1, parameter2,  ..... parameter n){
//function_body
}
return-type
```

The **return type** of a function is the data type of the variable that that function returns.

For eg - If we write a function that adds 2 integers and returns their sum then the return type of this function will be 'int' as we will return a sum that is an integer value.
When a function does not return any value, in that case the return type of the function is 'void'.

function_name
It is the unique name of that function.
It is always recommended to declare a function before it is used.

Parameters
A function can take some parameters as inputs. These parameters are specified along with their data types.
For eg- if we are writing a function to add 2 integers, the parameters would be passed like –
int add (int num1, int num2)

main function

The main function is a special function as the computer starts running the code from the beginning of the main function. Main function serves as the entry point for the program.

Example :

```
package com.apnacollege;

public class Main {
   //A METHOD to calculate sum of 2 numbers - a & b
   public static void sum(int a, int b) {
       int sum = a + b;
       System.out.println(sum);
   }

   public static void main(String[] args) {
       int a = 10;
       int b = 20;
       sum(a, b); // Function Call

   }
}
```

Qs. Write a function to multiply 2 numbers.

```
import java.util.*;


public class Functions {


   //Multiply 2 numbers
   public static int multiply(int a, int b) {

       return a*b;

   }

   public static void main(String args[]) {

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

Apna College

```
        int a = sc.nextInt();

        int b = sc.nextInt();



        int result = multiply(a, b);

        System.out.println(result);

    }

}
```

Qs. Write a function to calculate the factorial of a number.

```
import java.util.*;

public class Functions {
    // public static int calculateSum(int a, int b) {
    //     int sum = a + b;
    //     return sum;
    // }

    // public static int calculateProduct(int a, int b) {
    //     return a * b;
    // }

    public static void printFactorial(int n) {
        //loop
        if(n < 0) {
            System.out.println("Invalid Number");
            return;
        }
        int factorial = 1;

        for(int i=n; i>=1; i--) {
            factorial = factorial * i;
        }

        System.out.println(factorial);
        return;
    }
```

Apna College

```
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        printFactorial(n);
    }
}
```

Qs. Write a function to calculate the product of 2 numbers.

```
import java.util.*;



public class Functions {

    // public static int calculateSum(int a, int b) {

    //     int sum = a + b;

    //     return sum;

    // }



    public static int calculateProduct(int a, int b) {

       return a * b;

    }



        public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        int a = sc.nextInt();

        int b = sc.nextInt();

        System.out.println(calculateProduct(a, b));

    }

}
```

Apna College

Homework Problems

1. Make a function to check if a number is prime or not.
2. Make a function to check if a given number n is even or not.
3. Make a function to print the table of a given number n.
4. Read about Recursion.

Apna College

Java - Introduction to Programming

Exercise 1

Questions

1. Enter 3 numbers from the user & make a function to print their average.
2. Write a function to print the sum of all odd numbers from 1 to n.
3. Write a function which takes in 2 numbers and returns the greater of those two.
4. Write a function that takes in the radius as input and returns the circumference of a circle.
5. Write a function that takes in age as input and returns if that person is eligible to vote or not. A person of age > 18 is eligible to vote.
6. Write an infinite loop using do while condition.
7. Write a program to enter the numbers till the user wants and at the end it should display the count of positive, negative and zeros entered.
8. Two numbers are entered by the user, x and n. Write a function to find the value of one number raised to the power of another i.e. x^n.
9. Write a function that calculates the Greatest Common Divisor of 2 numbers. (BONUS)
10. Write a program to print Fibonacci series of n terms where n is input by user :
    0 1 1 2 3 5 8 13 21 .....
    In the Fibonacci series, a number is the sum of the previous 2 numbers that came before it.
    (BONUS)

Apna College

Java - Introduction to Programming
Exercise 1 SOLUTIONS

1. Enter 3 numbers from the user & make a function to print their average.
   //Try to convert it into a function on your own.

```
import java.util.*;

public class Solutions {
   public static void main(String args[]) {
      Scanner sc = new Scanner(System.in);
      int a = sc.nextInt();
      int b = sc.nextInt();
      int c = sc.nextInt();

      int average = (a + b + c) / 3;
      System.out.println(average);
   }
}
```

2. Write a function to print the sum of all odd numbers from 1 to n.

```
import java.util.*;

public class Solutions {
   public static void printSum(int n) {
       int sum = 0;

      for(int i=1; i<=n; i++) {
        if(i % 2 != 0) {
            sum = sum + i;
        }
      }

      System.out.println(sum);
   }
   public static void main(String args[]) {
      Scanner sc = new Scanner(System.in);
```

```
        int n = sc.nextInt();
        printSum(n);
    }
}
```

3. Write a function which takes in 2 numbers and returns the greater of those two.

```
import java.util.*;

public class Solutions {
    public static int getGreater(int a, int b) {
        if(a > b) {
            return a;
        } else {
            return b;
        }
    }
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();
        System.out.println(getGreater(a, b));
    }
}
```

4. Write a function that takes in the radius as input and returns the circumference of a circle.

```
import java.util.*;

public class Solutions {
    public static Double getCircumference(Double radius) {
        return 2 * 3.14 * radius;
    }
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        Double r = sc.nextDouble();
        System.out.println(getCircumference(radius));
    }
}
```

5. Write a function that takes in age as input and returns if that person is eligible to vote or not. A person of age > 18 is eligible to vote.

```
import java.util.*;

public class Solutions {
   public static boolean isElligible(int age) {
        if(age > 18) {
            return true;
        }
        return false;
   }
   public static void main(String args[]) {
      Scanner sc = new Scanner(System.in);
      int age = sc.nextInt();
      System.out.println(isElligible(age));
   }
}
```

6. Write an infinite loop using do while condition.

```
import java.util.*;

public class Solutions {
   public static void main(String args[]) {
      do {

      } while(true);
   }
}
```

7. Write a program to enter the numbers till the user wants and at the end it should display the count of positive, negative and zeros entered.

```
import java.util.*;

public class Solutions {
   public static void main(String args[]) {
        int positive = 0, negative = 0, zeros = 0;
        System.out.println("Press 1 to continue & 0 to stop");
        Scanner sc = new Scanner(System.in);
        int input = sc.nextInt();
```

```
        while(input == 1) {
            System.out.println("Enter your number : ");
            int number = sc.nextInt();
            if(number > 0) {
                positive++;
            } else if(number < 0) {
                negative++;
            } else {
                zeros++;
            }

            System.out.println("Press 1 to continue & 0 to stop");
            input = sc.nextInt();
        }

        System.out.println("Positives : "+ positive);
        System.out.println("Negatives : "+ negative);
        System.out.println("Zeros : "+ zeros);
    }
}
```

8. Two numbers are entered by the user, x and n. Write a function to find the value of one number raised to the power of another i.e. x^n.
   //Try to convert it into a function on your own.

```
import java.util.*;

public class Solutions {
   public static void main(String args[]) {
        System.out.println("Enter x");
        Scanner sc = new Scanner(System.in);
        int x = sc.nextInt();
        System.out.println("Enter n");
        int n = sc.nextInt();

        int result = 1;
        //Please see that n is not too large or else result will exceed the size
of int
        for(int i=0; i<n; i++) {
```

```
            result = result * x;
        }

        System.out.println("x to the power n is : "+ result);
    }
}
```

9. Write a function that calculates the Greatest Common Divisor of 2 numbers. (BONUS)

```
import java.util.*;

public class Solutions {
   public static void main(String args[]) {
       Scanner sc = new Scanner(System.in);
       int n1 = sc.nextInt();
       int n2 = sc.nextInt();

       while(n1 != n2) {
           if(n1>n2) {
               n1 = n1 - n2;
           } else {
               n2 = n2 - n1;
           }
       }
       System.out.println("GCD is : "+ n2);
   }
}
```

//Try to convert it into a function on your own.

10. Write a program to print Fibonacci series of n terms where n is input by user :
0 1 1 2 3 5 8 13 21 .....
In the Fibonacci series, a number is the sum of the previous 2 numbers that came before it.
(BONUS)

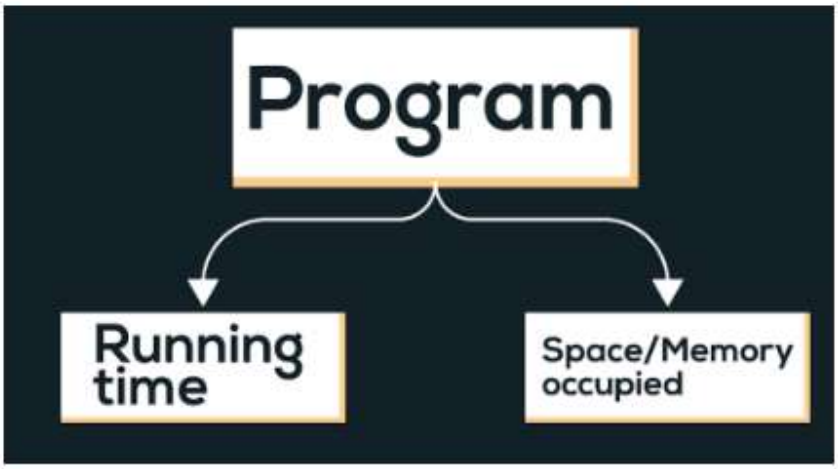
```
import java.util.*;

public class Solutions {
   public static void main(String args[]) {
       Scanner sc = new Scanner(System.in);
```

```
            int n = sc.nextInt();

            int a = 0, b = 1;

            System.out.print(a+" ");

            if(n > 1) {
                //find nth term
                for(int i=2; i<=n; i++) {
                    System.out.print(b+" ");
                    //the concept below is called swapping
                    int temp = b;
                    b = a + b;
                    a = temp;
                }

                System.out.println();
            }
        }
    }
```

Java - Introduction to Programming

Lecture 8

Time & Space Complexity



Time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the input.

Types of notations

1. O-notation: It is used to denote asymptotic upper bound. For a given function g(n), we denote it by O(g(n)). Pronounced as "big-oh of g of n". It is also known as worst case time complexity as it denotes the upper bound in which the algorithm terminates.
2. Ω-notation: It is used to denote asymptotic lower bound. For a given function g(n), we denote it by Ω(g(n)). Pronounced as "big-omega of g of n". It is also known as best case time complexity as it denotes the lower bound in which the algorithm terminates.
3. Θ-notation: It is used to denote the average time of a program.

Examples :

Apna College

```
int a = 0;
for (int i = 1; i <= n; i++)
{
  a = a + 1;
}
```

Linear Time Complexity. O(n)

Comparison of functions on the basis of time complexity

It follows the following order in case of time complexity:

$O(n^n) > O(n!) > O(n^3) > O(n^2) > O(n.log(n)) > O(n.log(log(n))) > O(n) > O(sqrt(n)) > O(log(n)) > O(1)$

Note: Reverse is the order for better performance of a code with corresponding time complexity, i.e. a program with less time complexity is more efficient.

Space Complexity

Space complexity of an algorithm quantifies the amount of time taken by a program to run as a function of length of the input. It is directly proportional to the largest memory your program acquires at any instance during run time.
For example: *int* consumes 4 bytes of memory.

Apna College

Java - Introduction to Programming
Lecture 10

Arrays In Java

Arrays in Java are like a list of elements of the same type i.e. a list of integers, a list of booleans etc.

a. Creating an Array (method 1) - with new keyword

```
int[] marks = new int[3];
marks[0] = 97;
marks[1] = 98;
marks[2] = 95;
```

b. Creating an Array (method 2)

```
int[] marks = {98, 97, 95};
```

c. Taking an array as an input and printing its elements.

```
import java.util.*;

public class Arrays {
   public static void main(String args[]) {
       Scanner sc = new Scanner(System.in);
       int size = sc.nextInt();
       int numbers[] = new int[size];

       for(int i=0; i<size; i++) {
           numbers[i] = sc.nextInt();
       }

       //print the numbers in array
       for(int i=0; i<arr.length; i++) {
           System.out.print(numbers[i]+" ");
       }
   }
}
```

Apna College

Homework Problems

1. Take an array of names as input from the user and print them on the screen.

```java
import java.util.*;



public class Arrays {

   public static void main(String args[]) {

      Scanner sc = new Scanner(System.in);

      int size = sc.nextInt();

      String names[] = new String[size];



      //input

      for(int i=0; i<size; i++) {

          names[i] = sc.next();

      }



      //output

       for(int i=0; i<names.length; i++) {

           System.out.println("name " + (i+1) +" is : " + names[i]);

       }



   }

}
```

2. Find the maximum & minimum number in an array of integers.

Apna College

[HINT : Read about Integer.MIN_VALUE & Integer.MAX_VALUE in Java]

```java
import java.util.*;



public class Arrays {

   public static void main(String args[]) {

      Scanner sc = new Scanner(System.in);

      int size = sc.nextInt();

      int numbers[] = new int[size];



      //input

      for(int i=0; i<size; i++) {

          numbers[i] = sc.nextInt();

      }



      int max = Integer.MIN_VALUE;

      int min = Integer.MAX_VALUE;



       for(int i=0; i<numbers.length; i++) {

           if(numbers[i] < min) {

               min = numbers[i];

           }

           if(numbers[i] > max) {

               max = numbers[i];

           }

       }
```

Apna College

```
        System.out.println("Largest number is : " + max);

        System.out.println("Smallest number is : " + min);



    }

}
```

3. Take an array of numbers as input and check if it is an array sorted in ascending order.

   Eg : { 1, 2, 4, 7 } is sorted in ascending order.

   {3, 4, 6, 2} is not sorted in ascending order.

```
import java.util.*;



public class Arrays {

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        int size = sc.nextInt();

        int numbers[] = new int[size];



        //input

        for(int i=0; i<size; i++) {

            numbers[i] = sc.nextInt();

        }



        boolean isAscending = true;
```

```
        for(int i=0; i<numbers.length-1; i++) { // NOTICE numbers.length - 1 as
termination condition

            if(numbers[i] > numbers[i+1]) { // This is the condition for
descending order

                isAscending = false;

            }

        }



        if(isAscending) {

            System.out.println("The array is sorted in ascending order");

        } else {

            System.out.println("The array is not sorted in ascending order");

        }



    }

}
```

Apna College

Java - Introduction to Programming
Lecture 11

2D Arrays In Java

It is similar to 2D matrices that we studied in 11th and 12th class.

a. Creating a 2D Array - with new keyword

```
int[][] marks = new int[3][3];
```

b. Taking a matrix as an input and printing its elements.

```
import java.util.*;

public class TwoDArrays {
   public static void main(String args[]) {
       Scanner sc = new Scanner(System.in);
       int rows = sc.nextInt();
       int cols = sc.nextInt();

       int[][] numbers = new int[rows][cols];

       //input
       //rows
       for(int i=0; i<rows; i++) {
           //columns
           for(int j=0; j<cols; j++) {
               numbers[i][j] = sc.nextInt();
           }
       }


       for(int i=0; i<rows; i++) {
           for(int j=0; j<cols; j++) {
                   System.out.print(numbers[i][j]+" ");
               }
               System.out.println();
           }
   }
```

Apna College

```
}
```

c. Searching for an element x in a matrix.

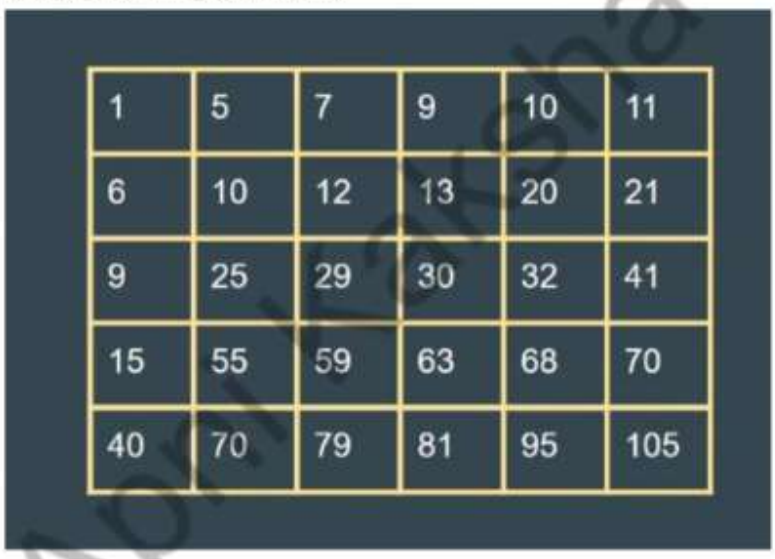
```
import java.util.*;

public class TwoDArrays {
   public static void main(String args[]) {
       Scanner sc = new Scanner(System.in);
       int rows = sc.nextInt();
       int cols = sc.nextInt();

       int[][] numbers = new int[rows][cols];

       //input
       //rows
       for(int i=0; i<rows; i++) {
           //columns
           for(int j=0; j<cols; j++) {
               numbers[i][j] = sc.nextInt();
           }
       }

       int x = sc.nextInt();

       for(int i=0; i<rows; i++) {
           for(int j=0; j<cols; j++) {
               //compare with x
               if(numbers[i][j] == x) {
                   System.out.println("x found at location (" + i + ", " + j +
")");
               }
           }
       }



   }
}
```

Apna College

Homework Problems

1. Print the spiral order matrix as output for a given matrix of numbers. [Difficult for Beginners]

For example: for the given matrix,



1	5	7	9	10	11
6	10	12	13	20	21
9	25	29	30	32	41
15	55	59	63	68	70
40	70	79	81	95	105

Spiral order is given by:
1 5 7 9 10 11 21 41 70 105 95 81 79 70 40 15 9 6 10 12 13 20 32 68 63 59 55 25 29 30 29.

APPROACH :

Algorithm: (We are given a 2D matrix of n X m).

1. We will need 4 variables:

a. row_start – initialized with 0.

b. row_end – initialized with n-1.

c. column_start – initialized with 0.

d. column_end – initialized with m-1.

2. First of all, we will traverse in the row row_start from column_start

to column_end and we will increase the row_start with 1 as we have traversed the starting row.

3. Then we will traverse in the column column_end from row_start to row_end and decrease the column_end by 1.

4. Then we will traverse in the row row_end from column_end to column_start and decrease the row_end by 1.

5. Then we will traverse in the column column_start from row_end to row_start and increase the column_start by 1.

6. We will do the above steps from 2 to 5 until row_start <= row_end and column_start <= column_end.

```
import java.util.*;


public class Arrays {

   public static void main(String args[]) {

      Scanner sc = new Scanner(System.in);

      int n = sc.nextInt();

      int m = sc.nextInt();


      int matrix[][] = new int[n][m];

      for(int i=0; i<n; i++) {

           for(int j=0; j<m; j++) {

               matrix[i][j] = sc.nextInt();

           }

      }


      System.out.println("The Spiral Order Matrix is : ");

      int rowStart = 0;
```

Apna College

```
        int rowEnd = n-1;
        int colStart = 0;
        int colEnd = m-1;

        //To print spiral order matrix
        while(rowStart <= rowEnd && colStart <= colEnd) {
            //1
            for(int col=colStart; col<=colEnd; col++) {
                System.out.print(matrix[rowStart][col] + " ");
            }
            rowStart++;

            //2
            for(int row=rowStart; row<=rowEnd; row++) {
                System.out.print(matrix[row][colEnd] +" ");
            }
            colEnd--;

            //3
            for(int col=colEnd; col>=colStart; col--) {
                System.out.print(matrix[rowEnd][col] + " ");
            }
            rowEnd--;

            //4
            for(int row=rowEnd; row>=rowStart; row--) {
```

Apna College

```
                System.out.print(matrix[row][colStart] + " ");

            }

            colStart++;



            System.out.println();

        }

    }

}
```

2. For a given matrix of N x M, print its transpose.

```
import java.util.*;



public class Arrays {

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();

        int m = sc.nextInt();



        int matrix[][] = new int[n][m];

        for(int i=0; i<n; i++) {

             for(int j=0; j<m; j++) {

                 matrix[i][j] = sc.nextInt();

             }

        }
```

Apna College

```
        System.out.println("The transpose is : ");
        //To print transpose
        for(int j=0; j<m ;j++) {
            for(int i=0; i<n; i++) {
                System.out.print(matrix[i][j]+" ");
            }
            System.out.println();
        }
    }
}
```

Apna College

Java - Introduction to Programming
Lecture 12

Strings

Declaration
```
String name = "Tony";
```

Taking Input
```
Scanner sc = new Scanner(System.in);
        String name = sc.next();
```

Concatenation (Joining 2 strings)
```
String firstName = "Tony";
        String secondName = "Stark";

        String fullName = firstName + " " + secondName;
        System.out.println(fullName);
```

Print length of a String
```
String firstName = "Tony";
        String secondName = "Stark";

        String fullName = firstName + " " + secondName;
        System.out.println(fullName.length());
```

Access characters of a string
```
String firstName = "Tony";
        String secondName = "Stark";

        String fullName = firstName + " " + secondName;

        for(int i=0; i<fullName.length(); i++) {
            System.out.println(fullName.charAt(i));
        }
```

Apna College

Compare 2 strings

```
import java.util.*;

public class Strings {
   public static void main(String args[]) {
       String name1 = "Tony";
       String name2 = "Tony";

       if(name1.equals(name2)) {
           System.out.println("They are the same string");
       } else {
           System.out.println("They are different strings");
       }

       //DO NOT USE == to check for string equality
       //Gives correct answer here
       if(name1 == name2) {
           System.out.println("They are the same string");
       } else {
           System.out.println("They are different strings");
       }

       //Gives incorrect answer here
       if(new String("Tony") == new String("Tony")) {
           System.out.println("They are the same string");
       } else {
           System.out.println("They are different strings");
       }

   }
}
```

Substring

The substring of a string is a subpart of it.

```
public class Strings {
   public static void main(String args[]) {
       String name = "TonyStark";

       System.out.println(name.substring(0, 4));
```

Apna College

```
    }
}
```

ParseInt Method of Integer class

```
public class Strings {
   public static void main(String args[]) {
       String str = "123";
       int number = Integer.parseInt(str);
       System.out.println(number);


   }
}
```

ToString Method of String class

```
public class Strings {
   public static void main(String args[]) {
      int number = 123;
      String str = Integer.toString(number);
      System.out.println(str.length());


   }
}
```

ALWAYS REMEMBER : Java Strings are Immutable.

Apna College

Homework Problems

1. Take an array of Strings input from the user & find the cumulative (combined) length of all those strings.

```
import java.util.*;



public class Strings {

   public static void main(String args[]) {

     Scanner sc = new Scanner (System.in);

     int size = sc.nextInt();

     String array[] = new String[size];

     int totLength = 0;



     for(int i=0; i<size; i++) {

       array[i] = sc.next();

       totLength += array[i].length();

     }



     System.out.println(totLength);

   }

}
```

2. Input a string from the user. Create a new string called ‘result’ in which you will replace the letter ‘e’ in the original string with letter ‘i’.

   Example :

   original = “eabcdef’ ; result = “iabcdif”

   Original = “xyz” ; result = “xyz”

Apna College

```java
import java.util.*;


public class Strings {

   public static void main(String args[]) {

     Scanner sc = new Scanner (System.in);

     String str = sc.next();

     String result = "";


     for(int i=0; i<str.length(); i++) {

       if(str.charAt(i) == 'e') {

         result += 'i';

       } else {

         result += str.charAt(i);

       }

     }


     System.out.println(result);

   }

}
```

3. Input an email from the user. You have to create a username from the email by deleting the part that comes after '@'. Display that username to the user.

   Example :

   email = "apnaCollegeJava@gmail.com" ; username = "apnaCollegeJava"

   email = "helloWorld123@gmail.com"; username = "helloWorld123"

```java
import java.util.*;


public class Strings {

   public static void main(String args[]) {

     Scanner sc = new Scanner (System.in);

     String email = sc.next();

     String userName = "";



     for(int i=0; i<email.length(); i++) {

       if(email.charAt(i) == '@') {

        break;

       } else {

         userName += email.charAt(i);

       }

     }



     System.out.println(userName);

   }

}
```

Apna College

Java - Introduction to Programming
Lecture 13

String Builder

Declaration

```
StringBuilder sb = new StringBuilder("Apna College");
     System.out.println(sb);
```

Get A Character from Index

```
StringBuilder sb = new StringBuilder("Tony");
     //Set Char
     System.out.println(sb.charAt(0));
```

Set a Character at Index

```
StringBuilder sb = new StringBuilder("Tony");
     //Get Char
     sb.setCharAt(0, 'P');
     System.out.println(sb);
```

Insert a Character at Some Index

```
import java.util.*;

public class Strings {
   public static void main(String args[]) {
     StringBuilder sb = new StringBuilder("tony");
     //Insert char
     sb.insert(0, 'S');
     System.out.println(sb);
   }
}
```

Apna College

Delete char at some Index

```
import java.util.*;

public class Strings {
   public static void main(String args[]) {
     StringBuilder sb = new StringBuilder("tony");
     //Insert char
     sb.insert(0, 'S');
     System.out.println(sb);

      //delete char
     sb.delete(0, 1);
     System.out.println(sb);
   }
}
```

Append a char

Append means to add something at the end.

```
import java.util.*;

public class Strings {
   public static void main(String args[]) {
     StringBuilder sb = new StringBuilder("Tony");
     sb.append(" Stark");
     System.out.println(sb);
   }
}
```

Print Length of String

```
import java.util.*;

public class Strings {
   public static void main(String args[]) {
     StringBuilder sb = new StringBuilder("Tony");
     sb.append(" Stark");
     System.out.println(sb);

     System.out.println(sb.length());
   }
}
```

Apna College

Reverse a String (using StringBuilder class)

```
import java.util.*;

public class Strings {
   public static void main(String args[]) {
     StringBuilder sb = new StringBuilder("HelloWorld");

     for(int i=0; i<sb.length()/2; i++) {
       int front = i;
       int back = sb.length() - i - 1;

       char frontChar = sb.charAt(front);
       char backChar = sb.charAt(back);

       sb.setCharAt(front, backChar);
       sb.setCharAt(back, frontChar);
     }

     System.out.println(sb);
   }
}
```

Homework Problems

Try Solving all the String questions with StringBuilder.

Apna College

Java - Introduction to Programming
Lecture 14

Bit Manipulation

Get Bit

```java
import java.util.*;

public class Bits {
   public static void main(String args[]) {
      int n = 5; //0101
      int pos = 3;
      int bitMask = 1<<pos;

      if((bitMask & n) == 0) {
          System.out.println("bit was zero");
      } else {
          System.out.println("bit was one");
      }
   }
}
```

Set Bit

```java
import java.util.*;

public class Bits {
   public static void main(String args[]) {
      int n = 5; //0101
      int pos = 1;
      int bitMask = 1<<pos;

      int newNumber = bitMask | n;
      System.out.println(newNumber);
   }
}
```

Apna College

Clear Bit

```java
import java.util.*;
public class Bits {
  public static void main(String args[]) {
     int n = 5; //0101
     int pos = 2;
     int bitMask = 1<<pos;
     int newBitMask = ~(bitMask);
     int newNumber = newBitMask & n;
     System.out.println(newNumber);
  }
}
```

Update Bit

```java
import java.util.*;

public class Bits {
   public static void main(String args[]) {
       Scanner sc = new Scanner(System.in);
       int oper = sc.nextInt();
       // oper=1 -> set; oper=0 -> clear
      int n = 5;
      int pos = 1;

      int bitMask = 1<<pos;
      if(oper == 1) {
          //set
          int newNumber = bitMask | n;
          System.out.println(newNumber);
      } else {
       //clear
       int newBitMask = ~(bitMask);
       int newNumber = newBitMask & n;
       System.out.println(newNumber);
      }

   }
}
```

Apna College

Homework Problems

1. Write a program to find if a number is a power of 2 or not.
2. Write a program to toggle a bit a position = “pos” in a number “n”.
3. Write a program to count the number of 1’s in a binary representation of the number.
4. Write 2 functions => decimalToBinary() & binaryToDecimal() to convert a number from one number system to another. [BONUS]

Apna College

Sorting in JAVA

1. **Bubble Sort**

   Idea: if arr[i] > arr[i+1] swap them. To place the element in their respective position, we have to do the following operation N-1 times.

   Time Complexity: $O(N^2)$

   Code

```java
import java.util.*;

class Sorting {
   public static void printArray(int arr[]) {
       for(int i=0; i<arr.length; i++) {
           System.out.print(arr[i]+" ");
       }
       System.out.println();
   }

   public static void main(String args[]) {
       int arr[] = {7, 8, 1, 3, 2};

       //bubble sort
       for(int i=0; i<arr.length-1; i++) {
           for(int j=0; j<arr.length-i-1; j++) {
               if(arr[j] > arr[j+1]) {
                   //swap
                   int temp = arr[j];
                   arr[j] = arr[j+1];
                   arr[j+1] = temp;
               }
           }
       }

       printArray(arr);
   }
```

```
}
```

2. Selection Sort

Idea: The inner loop selects the minimum element in the unsorted array and places the elements in increasing order.
Time complexity: $O(N^2)$

Code

```
import java.util.*;

class Sorting {
   public static void printArray(int arr[]) {
       for(int i=0; i<arr.length; i++) {
           System.out.print(arr[i]+" ");
       }
       System.out.println();
   }

   public static void main(String args[]) {
       int arr[] = {7, 8, 1, 3, 2};

       //selection sort
       for(int i=0; i<arr.length-1; i++) {
           int smallest = i;
           for(int j=i+1; j<arr.length; j++) {
               if(arr[j] < arr[smallest]) {
                   smallest = j;
               }
           }
           //swap
           int temp = arr[smallest];
           arr[smallest] = arr[i];
           arr[i] = temp;
       }

       printArray(arr);
   }
```

```
}
```

3. Insertion Sort

Idea: Take an element from the unsorted array, place it in its corresponding position in the sorted part, and shift the elements accordingly.

Time Complexity: $O(N^2)$

Code

```
import java.util.*;

class Sorting {
   public static void printArray(int arr[]) {
       for(int i=0; i<arr.length; i++) {
           System.out.print(arr[i]+" ");
       }
       System.out.println();
   }

   public static void main(String args[]) {
       int arr[] = {7, 8, 1, 3, 2};

       //insertion sort
       for(int i=1; i<arr.length; i++) {
           int current = arr[i];
           int j = i - 1;
               while(j >= 0 && arr[j] > current) {
                   //Keep swapping
                   arr[j+1] = arr[j];
                   j--;
               }
           arr[j+1] = current;
       }
       printArray(arr);
   }
}
```

JAVA Class 1 (Codes)

Q1. Print numbers from 5 to 1.

```
public static void printNumbers(int n) {
        if(n == 0) {
            return;
        }
        System.out.println(n);
        printNumbers(n-1);
    }
```

Q2. Print numbers from 1 to 5.

```
public static void printNumbers(int n) {
        if(n == 6) {
            return;
        }
        System.out.println(n);
        printNumbers(n+1);
    }
```

Q3. Print the sum of first n natural numbers.

```
class Recursion1 {
    public static void printSum(int n, int sum) {
        if(n == 0) {
            System.out.println(sum);
            return;
        }

        sum += n;
        printSum(n-1, sum);
    }
    public static void main(String args[]) {
        printSum(5, 0);
    }
}
```

Q4. Print factorial of a number n.

```
class Recursion1 {
   public static void printFactorial(int n, int fact) {
       if(n == 0) {
           System.out.println(fact);
           return;
       }

       fact *= n;
       printFactorial(n-1, fact);
   }
   public static void main(String args[]) {
       printFactorial(5, 1);
   }
}
```

Q5. Print the fibonacci sequence till nth term.

```
class Recursion1 {
   public static void printFactorial(int a, int b, int n) {
       if(n == 0) {
           return;
       }

       System.out.println(a);
       printFactorial(b, a+b, n-1);
   }
   public static void main(String args[]) {
       printFactorial(0, 1, 5);
   }
}
```

Q6. Print x^n (with stack height = n)

```
class Recursion1 {
   public static int printPower(int x, int n) {
       if(n == 0) {
           return 1;
       }

       if(x == 0) {
           return 0;
       }
```

```
        int x_ = printPower(x, n-1);
        int xn = x * x_;
        return xn;
    }
    public static void main(String args[]) {
        int x = 2, n = 5;
        int output = printPower(x, n);
        System.out.println(output);
    }
}
```

Q7. Print x^n (with stack height = logn)

```
class Recursion1 {
    public static int printPower(int x, int n) {
        if(n == 0) {
            return 1;
        }

        if(n % 2 == 0) {
            return printPower(x, n/2) * printPower(x, n/2);
        }

        else {
            return x * printPower(x, n/2) * printPower(x, n/2);
        }
    }

    public static void main(String args[]) {
        int x = 2, n = 5;
        int output = printPower(x, n);
        System.out.println(output);
    }
}
```

Recursion Class 2

Q1. Tower of Hanoi - Transfer n disks from source to destination over 3 towers.

```
public class Recursion2 {
   public static void towerOfHanoi(int n, String src, String helper, String dest) {
       if(n == 1) {
           System.out.println("transfer disk " + n + " from " + src + " to " + dest);
           return;
       }

       //transfer top n-1 from src to helper using dest as 'helper'
       towerOfHanoi(n-1, src, dest, helper);

       //transfer nth from src to dest
       System.out.println("transfer disk " + n + " from " + src + " to " + helper);

       //transfer n-1 from helper to dest using src as 'helper'
       towerOfHanoi(n-1, helper, src, dest);

   }
   public static void main(String args[]) {
       int n = 4;
       towerOfHanoi(n, "A", "B", "C");
   }
}
```

Q2. Print a string in reverse.

```
public class Recursion2 {
   public static String revString(String str) {
       if(str.length() == 1) {
           return str;
       }

       char currChar = str.charAt(0);
       String nextString = revString(str.substring(1));
       return nextString + currChar;
   }
   public static void main(String args[]) {
```

```
        String str = "abcd";
        String reversed = revString(str);
        System.out.println(reversed);
    }
}
```

Q3. Find the occurrence of the first and last occurrence of an element using recursion.

```
public class Recursion2 {
    public static int first = -1;
    public static int last = -1;

    public static void getIndices(String str, char el, int idx) {
        if(idx == str.length()) {
            return;
        }

        if(str.charAt(idx) == el) {
            if(first == -1) {
                first = idx;
            } else {
                last = idx;
            }
        }
        getIndices(str, el, idx+1);
    }
    public static void main(String args[]) {
        String str = "tabcdfghijakkk";
        char el = 'a';
        getIndices(str, el, 0);
        System.out.println("First occurence : " + first);
        System.out.println("Last occurence : " + last);
    }
}
```

Q4. Check if an array is sorted (strictly increasing). - O(n)

```
public class Recursion2 {
    public static boolean checkIfIncreasing(int arr[], int idx) {
```

```
        if(idx == arr.length-1) {
            return true;
        }

        if(!checkIfIncreasing(arr, idx+1)) {
            return false;
        }
        return arr[idx] < arr[idx + 1];
    }

    public static void main(String args[]) {
        int arr1[] = {1, 2, 3, 4, 5};
        int arr2[] = {1, 6, 3, 4, 5};

        if(checkIfIncreasing(arr2, 0)) {
            System.out.println("Strictly Increasing");
        } else {
            System.out.println("NOT Strictly Increasing");
        }
    }
}
```

Q5. Move all ‘x’ to the end of the string. - O(n)

```
public class Recursion2 {
    //to add all 'x' to the end of the string
    public static String addX(int count) {
        String newStr = "x";
            for(int i=1;i<count; i++) {
                newStr += 'x';
            }
        return newStr;
    }

    public static String moveAllX(String str, int idx, int count) {
        if(idx == str.length()) {
            return addX(count);
        }

        if(str.charAt(idx) == 'x') {
```

```
            return moveAllX(str, idx+1, count+1);
        } else {
            String nextStr = moveAllX(str, idx+1, count);
            return str.charAt(idx) + nextStr;
        }
    }

    public static void main(String args[]) {
        String str = "abcdefxghxixjxxxk";
        int count = 0;

        String newStr = moveAllX(str, 0, count);
        System.out.println(newStr);
    }
}
```

Q6. Remove duplicates in a string.

```
public class Recursion2 {
    public static String removeDuplicates(String str, int idx, boolean present[]) {
        if(idx == str.length()) {
            return "";
        }

        char curr = str.charAt(idx);
        if(present[curr-'a']) {
            return removeDuplicates(str, idx+1, present);
        } else {
            present[curr-'a'] = true;
            return curr + removeDuplicates(str, idx+1, present);
        }
    }
    public static void main(String args[]) {
        String str = "abcadbcefghabi";
        boolean present[] = new boolean[str.length()];
        System.out.println(removeDuplicates(str, 0, present));
    }
}
```

Q7. Print all the subsequences of a string.

```
public class Recursion2 {
   public static void printSubseq(String str, int idx, String res) {
       if(idx == str.length()) {
           System.out.println(res);
           return;
       }

       //choose
       printSubseq(str, idx+1, res+str.charAt(idx));

       //don't choose
       printSubseq(str, idx+1, res);
   }
   public static void main(String args[]) {
       String str1 = "abc";
       String str2 = "aaa";

       printSubseq(str1, 0, "");
   }
}
```

Time complexity - O(2^n)

Q8. Print all unique subsequences of a string.

```
import java.util.HashSet;

public class Recursion2 {
   public static void printSubseq(String str, int idx, String res, HashSet<String>
allSubseq) {
       if(idx == str.length()) {
           if(allSubseq.contains(res)) {
               return;
           }
           allSubseq.add(res);
           System.out.println(res);
           return;
       }
```

```
        //choose
        printSubseq(str, idx+1, res+str.charAt(idx), allSubseq);

        //don't choose
        printSubseq(str, idx+1, res, allSubseq);
    }
    public static void main(String args[]) {
        String str1 = "abc";
        String str2 = "aaa";
        HashSet<String> allSubseq = new HashSet<>();
        printSubseq(str2, 0, "", allSubseq);
    }
}
```

Q9. Print keypad combination

(0 -> .;
1 -> abc
2 -> def
3 -> ghi
4 -> jkl
5 -> mno
6 -> pqrs
7 -> tu
8 -> vwx
9 -> yz
)

```
import java.util.HashSet;

public class Recursion2 {
    public static String keypad[] = {".", "abc", "def", "ghi", "jkl", "mno", "pqrs",
"tu", "vwx", "yz"};

    public static void printKeypadCombination(String number, int idx, String res) {
        if(idx == number.length()) {
            System.out.println(res);
            return;
        }
```

```
        for(int i=0; i<keypad[number.charAt(idx)-'0'].length(); i++) {
            char curr = keypad[number.charAt(idx)-'0'].charAt(i);
            printKeypadCombination(number, idx+1, res+curr);
        }
    }
    public static void main(String args[]) {
        String number = "23";
        printKeypadCombination(number, 0, "");
    }
}
```

ADVANCED

Q1. Print all the permutations of a string.

```
public class Recursion3 {

    public static void printPermutation(String str, int idx, String perm) {
        if(str.length() == 0) {
            System.out.println(perm);
            return;
        }

        for(int i=0; i<str.length(); i++) {
            char currChar = str.charAt(i);
            String newStr = str.substring(0, i) + str.substring(i+1);
            printPermutation(newStr, idx+1, perm+currChar);
        }
    }
    public static void main(String args[]) {
        String str = "abc";
        printPermutation(str, 0, "");
    }
}
```

Time complexity - O(n*n!)

Q2. CountPathMaze

```
public class Recursion3 {

    public static int countPaths(int i, int j, int m, int n) {
        if(i == m-1 || j == n-1) {
            return 1;
        }

        return countPaths(i+1, j, m, n) + countPaths(i, j+1, m, n);
    }

    public static void main(String args[]) {
        int m = 4, n = 5;
        System.out.println(countPaths(0, 0, m, n));
    }
}
```

```

```

Time complexity - O(2^(m+n))

Q3. Tiling problem

```
public class Recursion3 {

   public static int placeTiles(int n, int m) {
       if(n < m) {
           return 1;
       } else if(n == m) {
           return 2;
       }

       return placeTiles(n-1, m) + placeTiles(n-m, m);
   }

   public static void main(String args[]) {
       int n = 4, m = 4;
       System.out.println(placeTiles(n, m));
   }
}
```

Q4. Friends pairing problem

```
public class Recursion3 {

   public static int pairFriends(int n) {
      if(n <= 1) {
          return 1;
      }

       return pairFriends(n-1) + (n-1) * pairFriends(n-2);
   }

   public static void main(String args[]) {
       int n = 3;
       System.out.println(pairFriends(n));
   }
}
```

Q5. Subsets of a set

```
import java.util.ArrayList;

public class Recursion3 {

   public static void printSubsets(ArrayList<Integer> subset) {
       for(int i=0; i<subset.size(); i++) {
           System.out.print(subset.get(i)+" ");
       }
       System.out.println();
   }

   public static void findSubsets(int n, ArrayList<Integer> subset) {
       if(n == 0) {
           printSubsets(subset);
           return;
       }

       findSubsets(n-1, subset);
       subset.add(n);
       findSubsets(n-1, subset);
       subset.remove(subset.size() - 1);
   }

   public static void main(String args[]) {
       int n = 3;
       findSubsets(n, new ArrayList<Integer> ());
   }
}
```

Backtracking in Recursion
Java

Print all Permutations

Time complexity - O(n*n!)

```java
public class Recursion3 {

    public static void printPermutation(String str, int idx, String perm) {
        if(str.length() == 0) {
            System.out.println(perm);
            return;
        }

        for(int i=0; i<str.length(); i++) {
            char currChar = str.charAt(i);
            String newStr = str.substring(0, i) + str.substring(i+1);
            printPermutation(newStr, idx+1, perm+currChar);
        }
    }
    public static void main(String args[]) {
        String str = "abc";
        printPermutation(str, 0, "");
    }
}
```

N-Queens

Time complexity - O(n^n)

```java
class Solution {

    public boolean isSafe(int row, int col, char[][] board) {

        //horizontal
```

Apna College

```
        for(int j=0; j<board.length; j++) {
            if(board[row][j] == 'Q') {
                return false;
            }
        }

        //vertical
        for(int i=0; i<board.length; i++) {
            if(board[i][col] == 'Q') {
                return false;
            }
        }

        //upper left
        int r = row;
        for(int c=col; c>=0 && r>=0; c--, r--) {
            if(board[r][c] == 'Q') {
                return false;
            }
        }

        //upper right
        r = row;
        for(int c=col; c<board.length && r>=0; r--, c++) {
            if(board[r][c] == 'Q') {
                return false;
```

Apna College

```
            }
        }

        //lower left
        r = row;
        for(int c=col; c>=0 && r<board.length; r++, c--) {
            if(board[r][c] == 'Q') {
                return false;
            }
        }

        //lower right
        for(int c=col; c<board.length && r<board.length; c++, r++) {
            if(board[r][c] == 'Q') {
                return false;
            }
        }

        return true;
    }

    public void saveBoard(char[][] board, List<List<String>> allBoards) {
        String row = "";
        List<String> newBoard = new ArrayList<>();

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

Apna College

```
            row = "";
            for(int j=0; j<board[0].length; j++) {
                if(board[i][j] == 'Q')
                    row += 'Q';
                else
                    row += '.';
            }
            newBoard.add(row);
        }

        allBoards.add(newBoard);
    }

    public void helper(char[][] board, List<List<String>> allBoards, int col) {
        if(col == board.length) {
            saveBoard(board, allBoards);
            return;
        }

        for(int row=0; row<board.length; row++) {
            if(isSafe(row, col, board)) {
                board[row][col] = 'Q';
                helper(board, allBoards, col+1);
                board[row][col] = '.';
            }
        }
```

Apna College

```
    }

    public List<List<String>> solveNQueens(int n) {
        List<List<String>> allBoards = new ArrayList<>();
        char[][] board = new char[n][n];

        helper(board, allBoards, 0);
        return allBoards;
    }
}
```

Homework Problems

1. https://leetcode.com/problems/permutations/ (Similar to print Permutations)
2. https://www.hackerrank.com/challenges/knightl-on-chessboard/problem (Similar to N-Queens)
3. https://leetcode.com/problems/sudoku-solver/ (Will be discussed in next class)

Apna College

Sudoku Solver (Backtracking 2)

Java

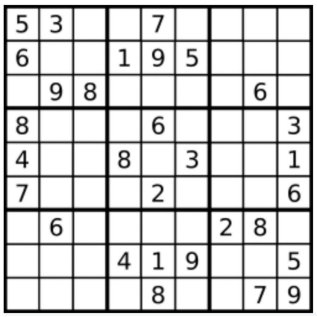
Write a program to solve a Sudoku puzzle by filling the empty cells.

A sudoku solution must satisfy all of the following rules:

1. Each of the digits 1-9 must occur exactly once in each row.
2. Each of the digits 1-9 must occur exactly once in each column.
3. Each of the digits 1-9 must occur exactly once in each of the 9 3x3 sub-boxes of the grid.

The '.' character indicates empty cells.

Sample Input



```
board =
[["5","3",".",".","7",".",".",".","."],["6",".",".","1","9","5",".",".","."],[
".","9","8",".",".",".",".","6","."],["8",".",".",".","6",".",".",".","3"],["4
",".",".","8",".","3",".",".","1"],["7",".",".",".","2",".",".",".","6"],[".",
"6",".",".",".",".","2","8","."],[".",".",".","4","1","9",".",".","5"],[".","."
,".",".","8",".",".","7","9"]]
```

Apna College

Sample Output

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

```
[["5","3","4","6","7","8","9","1","2"],["6","7","2","1","9","5","3","4","8"],[
"1","9","8","3","4","2","5","6","7"],["8","5","9","7","6","1","4","2","3"],["4
","2","6","8","5","3","7","9","1"],["7","1","3","9","2","4","8","5","6"],["9",
"6","1","5","3","7","2","8","4"],["2","8","7","4","1","9","6","3","5"],["3","4
","5","2","8","6","1","7","9"]]
```

Code

a. StartingRow = 3*(row/3) & StartingCol = 3*(col/3)
b. StartingRow = row - row%3 & StartingCol = col - col%3

Code

```
class Solution {
   public boolean isSafe(char[][] board, int row, int col, int number) {
       //column
       for(int i=0; i<board.length; i++) {
           if(board[i][col] == (char)(number+'0')) {
               return false;
           }
       }

       //row
       for(int j=0; j<board.length; j++) {
           if(board[row][j] == (char)(number+'0')) {
               return false;
           }
       }

       //grid
       int sr = 3 * (row/3);
       int sc = 3 * (col/3);

       for(int i=sr; i<sr+3; i++) {
           for(int j=sc; j<sc+3; j++) {
               if(board[i][j] == (char)(number+'0')) {
                   return false;
               }
           }
       }
```

Apna College

```
        return true;
    }

    public boolean helper(char[][] board, int row, int col) {
        if(row == board.length) {
            return true;
        }

        int nrow = 0;
        int ncol = 0;

        if(col == board.length-1) {
            nrow = row + 1;
            ncol = 0;
        } else {
            nrow = row;
            ncol = col + 1;
        }

        if(board[row][col] != '.') {
            if(helper(board, nrow, ncol)) {
                return true;
            }
        } else {

            //fill the place
            for(int i=1; i<=9; i++) {
                if(isSafe(board, row, col, i)) {
                    board[row][col] = (char)(i+'0');
                    if(helper(board, nrow, ncol))
                        return true;
                    else
                         board[row][col] = '.';
                }
            }
        }

        return false;
    }

    public void solveSudoku(char[][] board) {
        helper(board, 0, 0);
    }
}
```

Apna College

Code

```java
public static void divide(int arr[], int si, int ei) {
    if(si >= ei) {
        return;
    }

    int mid = si + (ei-si)/2;
    divide(arr, si, mid);
    divide(arr, mid+1, ei);
    conquer(arr, si, mid, ei);
}
```

```java
public static void conquer(int arr[], int si, int mid, int ei) {
    int merged[] = new int[ei-si+1];
    int idx1 = si;
    int idx2 = mid+1;
    int x = 0;
    while(idx1 <= mid && idx2 <= ei) {
        if(arr[idx1] <= arr[idx2]) {
            merged[x++] = arr[idx1++];
        } else {
            merged[x++] = arr[idx2++];
        }
    }

    while(idx1 <= mid) {
        merged[x++] = arr[idx1++];
    }

    while(idx2 <= ei) {
        merged[x++] = arr[idx2++];
    }

    for(int i=0, j=si; i<merged.length; i++, j++) {
        arr[j] = merged[i];
    }
}
```

Time Complexity : O(nlogn)

Code

```java
public static void quickSort(int arr[], int low, int high) {
    if (low < high) {
        int pi = partition(arr, low, high);

        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
```

```java
public static int partition(int[] arr, int low, int high) {
    int pivot = arr[high];
    int i = low-1;

    for(int j=low; j<high; j++) {
        if (arr[j] < pivot) {
            i++;
            //swap
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    //swap with pivot
    i++;
    int temp = arr[i];
    arr[i] = arr[high];
    arr[high] = temp;
    return i;
}
```

Time Complexity

Worst : O(n^2)

Average : O(nlogn)

Important

Worst case occurs when pivot is always the smallest or the largest element.

OBJECT ORIENTED PROGRAMMING SYSTEMS
JAVA

Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts defined below :

Class is a user-defined data type which defines its properties and its functions. Class is the only logical representation of the data. For example, Human being is a class. The body parts of a human being are its properties, and the actions performed by the body parts are known as functions. The class does not occupy any memory space till the time an object is instantiated.

Object is a run-time entity. It is an instance of the class. An object can represent a person, place or any other item. An object can operate on both data members and member functions.

Example 1:

```
class Student {
   String name;
   int age;

   public void getInfo() {
       System.out.println("The name of this Student is " + this.name);
       System.out.println("The age of this Student is " + this.age);
   }
}

public class OOPS {
   public static void main(String args[]) {
       Student s1 = new Student();
       s1.name = "Aman";
       s1.age = 24;
       s1.getInfo();

       Student s2 = new Student();
```

```
        s2.name = "Shradha";
        s2.age = 22;
        s2.getInfo();
    }
}
```

Example 2:

```
class Pen {
    String color;

    public void printColor() {
        System.out.println("The color of this Pen is " + this.color);
    }
}

public class OOPS {
    public static void main(String args[]) {
        Pen p1 = new Pen();
        p1.color = blue;

        Pen p2 = new Pen();
        p2.color = black;

        Pen p3 = new Pen();
        p3.color = red;

        p1.printColor();
        p2.printColor();
        p3.printColor();
    }
}
```

Note : When an object is created **using** a new keyword, then space is allocated for the variable in a heap, and the starting address is stored in the stack memory.

'this' keyword : 'this' keyword in Java that refers to the current instance of the class. In OOPS it is used to:

1. pass the current object as a parameter to another method
2. refer to the current class instance variable

Constructor : Constructor is a special method which is invoked automatically at the time of object creation. It is used to initialize the data members of new objects generally.

- Constructors have the same name as class or structure.
- Constructors don't have a return type. (Not even void)
- Constructors are only called once, at object creation.

There can be **three types** of constructors in Java.

1. Non-Parameterized constructor : A constructor which has no argument is known as non-parameterized constructor(or no-argument constructor). It is invoked at the time of creating an object. If we don't create one then it is created by default by Java.

```
class Student {
   String name;
   int age;

   Student() {
       System.out.println("Constructor called");
   }
}
```

2. Parameterized constructor : Constructor which has parameters is called a parameterized constructor. It is used to provide different values to distinct objects.

```
class Student {
   String name;
   int age;


   Student(String name, int age) {
       this.name = name;
       this.age = age;
   }
}
```

3. Copy Constructor : A Copy constructor is an **overloaded** constructor used to declare and initialize an object from another

object. There is only a user defined copy constructor in Java(C++ has a default one too).

```
class Student {
   String name;
   int age;

   Student(Student s2) {
       this.name = s2.name;
       this.age = s2.age;
   }
}
```

Note : Unlike languages like C++, Java has no Destructor. Instead, Java has an efficient garbage collector that deallocates memory automatically.

Polymorphism

Polymorphism is the ability to present the same interface for differing underlying forms (data types). With polymorphism, each of these classes will have different underlying data. Precisely, Poly means 'many' and morphism means 'forms'.

Types of Polymorphism IMP

1. Compile Time Polymorphism (Static)
2. Runtime Polymorphism (Dynamic)

Let's understand them one by one :

Compile Time Polymorphism : The polymorphism which is implemented at the compile time is known as compile-time polymorphism. Example – Method Overloading

Method Overloading : Method overloading is a technique which allows you to have more than one function with the same function name but with different functionality. Method overloading can be possible on the following basis:

1. The type of the parameters passed to the function.
2. The number of parameters passed to the function.

```
class Student {
   String name;
   int age;

   public void displayInfo(String name) {
       System.out.println(name);
   }

   public void displayInfo(int age) {
       System.out.println(age);
   }

   public void displayInfo(String name, int age) {
```

```
        System.out.println(name);
        System.out.println(age);
    }
}
```

Runtime Polymorphism : Runtime polymorphism is also known as **dynamic polymorphism**. Function overriding is an example of runtime polymorphism. Function overriding means when the child class contains the method which is already present in the parent class. Hence, **the child class overrides the method of the parent class.** In case of function overriding, parent and child classes both contain the same function with a different definition. The call to the function is determined at runtime is known as runtime polymorphism.

```
class Shape {
    public void area() {
        System.out.println("Displays Area of Shape");
    }
}
class Triangle extends Shape {
    public void area(int h, int b) {
        System.out.println((1/2)*b*h);
    }
}
class Circle extends Shape {
    public void area(int r) {
        System.out.println((3.14)*r*r);
    }
}
```

Inheritance

Inheritance is a process in which one object acquires all the properties and

behaviors of its parent object automatically. In such a way, you can **reuse, extend or modify** the attributes and behaviors which are defined in other classes.

In Java, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class.

Types of Inheritance :

1. Single inheritance : When one class inherits another class, it is known as single level inheritance

```
class Shape {
   public void area() {
        System.out.println("Displays Area of Shape");
   }
}
class Triangle extends Shape {
   public void area(int h, int b) {
        System.out.println((1/2)*b*h);
   }
}
```

2. Hierarchical inheritance : Hierarchical inheritance is defined as the process of deriving more than one class from a base class.

```
class Shape {
   public void area() {
        System.out.println("Displays Area of Shape");
   }
}
class Triangle extends Shape {
   public void area(int h, int b) {
        System.out.println((1/2)*b*h);
   }
}
class Circle extends Shape {
   public void area(int r) {
        System.out.println((3.14)*r*r);
   }
```

```
}
```

3. Multilevel inheritance : Multilevel inheritance is a process of deriving a class from another derived class.

```
class Shape {
   public void area() {
       System.out.println("Displays Area of Shape");
   }
}
class Triangle extends Shape {
   public void area(int h, int b) {
       System.out.println((1/2)*b*h);
   }
}
class EquilateralTriangle extends Triangle {
   int side;
}
```

4. Hybrid inheritance : Hybrid inheritance is a combination of simple, multiple inheritance and hierarchical inheritance.

Package in Java

Package is a group of similar types of classes, interfaces and sub-packages. Packages can be built-in or user defined.

Built-in packages - java, util, io etc.

```
import java.util.Scanner;
```

```
import java.io.IOException;
```

Access Modifiers in Java

- **Private**: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
- **Default**: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
- **Protected**: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
- **Public**: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.

```
package newpackage;


class Account {
   public String name;
   protected String email;
   private String password;


   public void setPassword(String password) {
       this.password = password;
   }
```

```
}
public class Sample {
   public static void main(String args[]) {
       Account a1 = new Account();
       a1.name = "Apna College";
       a1.setPassword("abcd");
       a1.email = "hello@apnacollege.com";
   }
}
```

Encapsulation

Encapsulation is the process of combining data and functions into a single unit called class. In Encapsulation, the data is not accessed directly; it is accessed through the functions present inside the class. In simpler words, attributes of the class are kept private and public getter and setter methods are provided to manipulate these attributes. Thus, encapsulation makes the concept of data hiding possible.(**Data hiding**: a language feature to restrict access to members of an object, reducing the negative effect due to dependencies. e.g. "protected", "private" feature in Java).

Abstraction

We try to obtain an **abstract view**, model or structure of a real life problem, and reduce its unnecessary details. With definition of properties of problems, including the data which are affected and the operations which are identified, the model abstracted from problems can be a standard solution to this type of problems. It is an efficient way since there are nebulous real-life problems that have similar properties.

In simple terms, it is hiding the unnecessary details & showing only the essential parts/functionalities to the user.

Data binding **:** Data binding is a process of binding the application UI and business logic. Any change made in the business logic will reflect directly to the application UI.

Abstraction is achieved in 2 ways :

- Abstract class
- Interfaces (Pure Abstraction)

1. **Abstract Class**

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.
- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.

```
abstract class Animal {
   abstract void walk();
   void breathe() {
```

```
        System.out.println("This animal breathes air");
    }
    Animal() {
        System.out.println("You are about to create an Animal.");
    }
}

class Horse extends Animal {
    Horse() {
        System.out.println("Wow, you have created a Horse!");
    }
    void walk() {
        System.out.println("Horse walks on 4 legs");
    }
}

class Chicken extends Animal {
    Chicken() {
        System.out.println("Wow, you have created a Chicken!");
    }
    void walk() {
        System.out.println("Chicken walks on 2 legs");
    }
}

public class OOPS {
    public static void main(String args[]) {
       Horse horse = new Horse();
       horse.walk();
       horse.breathe();
    }
}
```

2. **Interfaces**

- All the fields in interfaces are public, static and final by default.
- All methods are public & abstract by default.
- A class that implements an interface must implement all the methods declared in the interface.
- Interfaces support the functionality of multiple inheritance.

```
interface Animal {

   void walk();

}



class Horse implements Animal {

   public void walk() {

       System.out.println("Horse walks on 4 legs");

   }

}



class Chicken implements Animal {

   public void walk() {

       System.out.println("Chicken walks on 2 legs");

   }

}
```

```
public class OOPS {

   public static void main(String args[]) {

      Horse horse = new Horse();

      horse.walk();

   }

}
```

Static Keyword

Static can be :

1. Variable (also known as a class variable)
2. Method (also known as a class method)
3. Block
4. Nested class

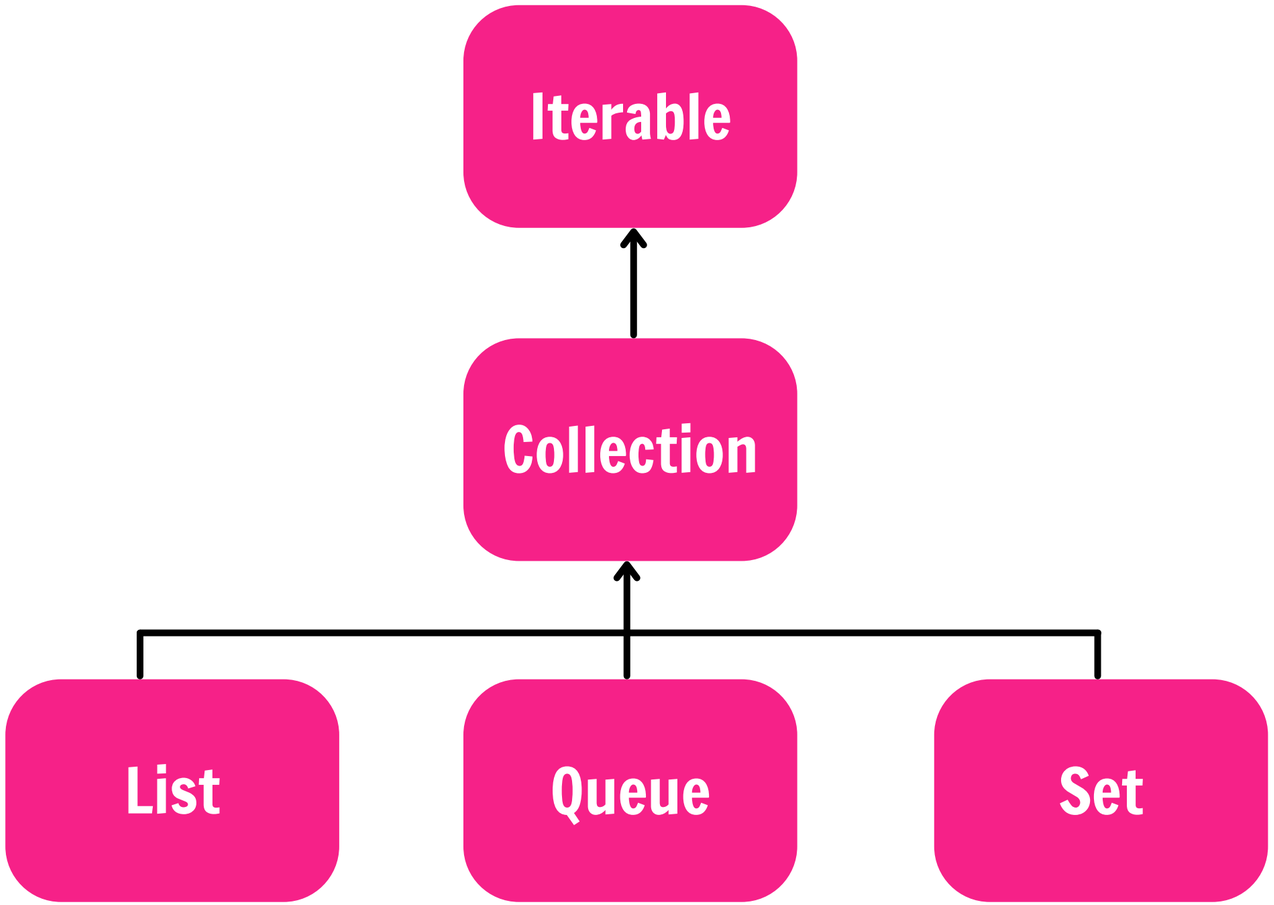
```
class Student {

   static String school;

   String name;

}

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

```
        Student.school = "JMV";
        Student s1 = new Student();
        Student s2 = new Student();

        s1.name = "Meena";
        s2.name = "Beena";

        System.out.println(s1.school);
        System.out.println(s2.school);
    }
}
```

JAVA

Collection Framework

Collection of Classes & Interfaces

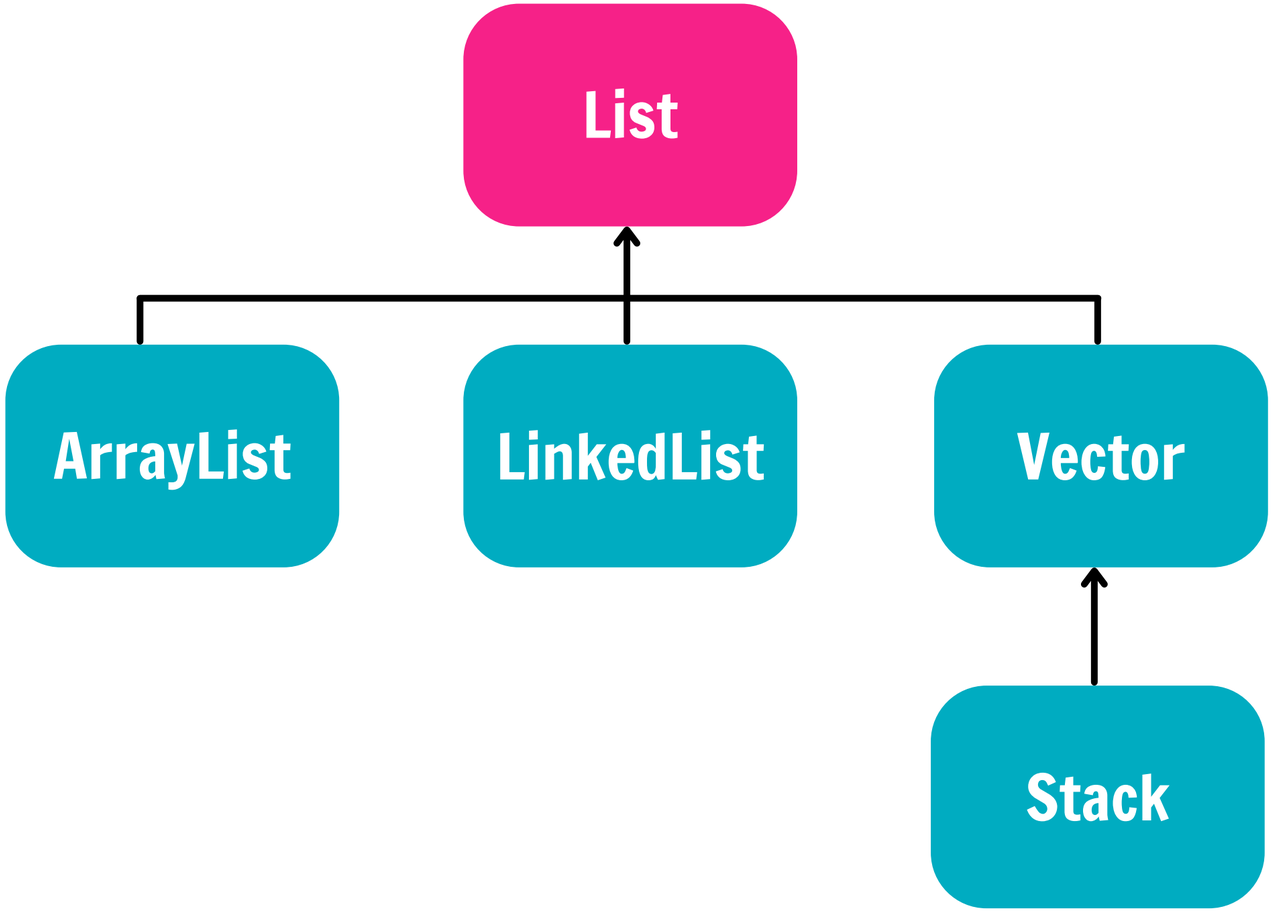


JAVA

Methods on Collections

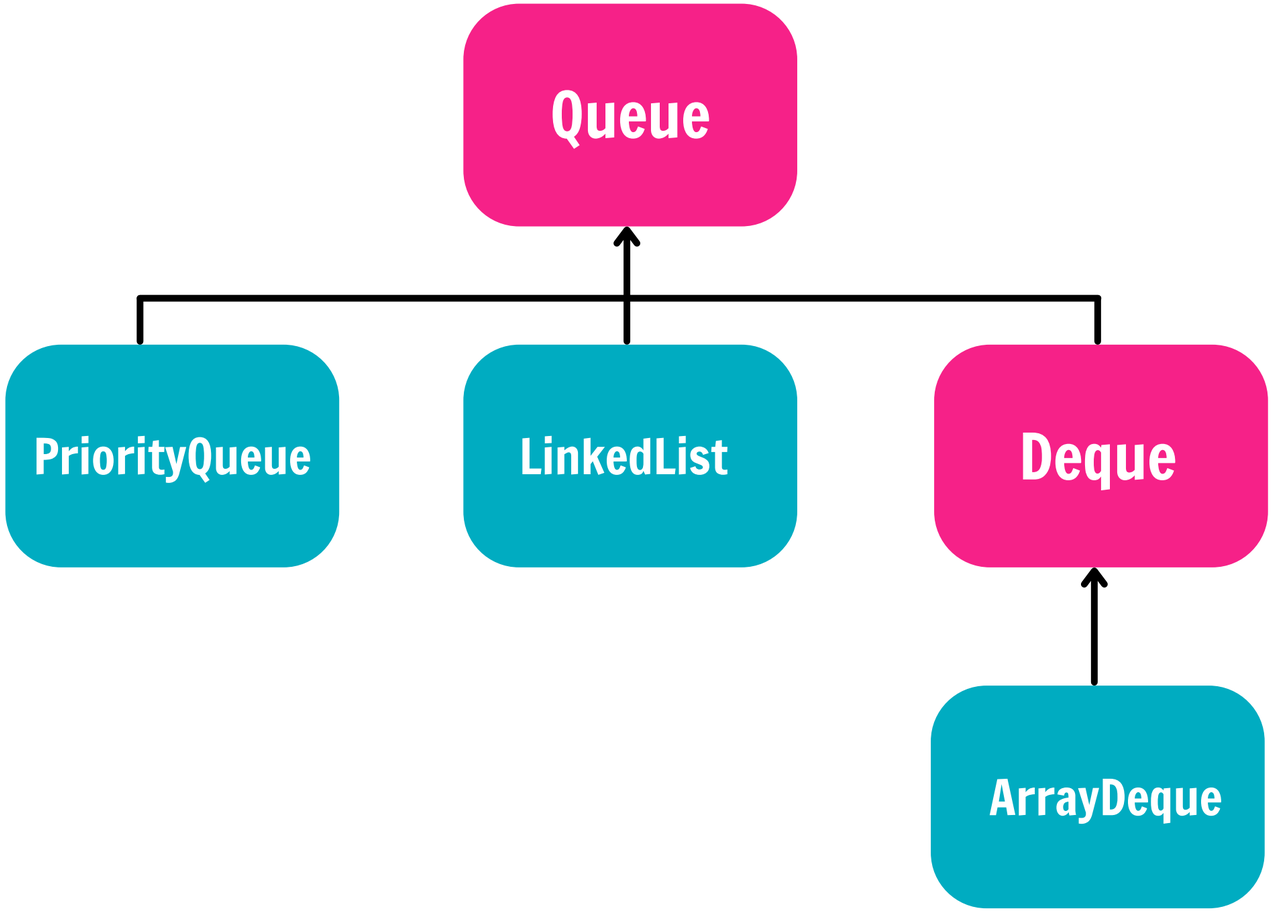
- add
- size
- remove
- iterate
- addAll
- removeAll
- clear

List Interface

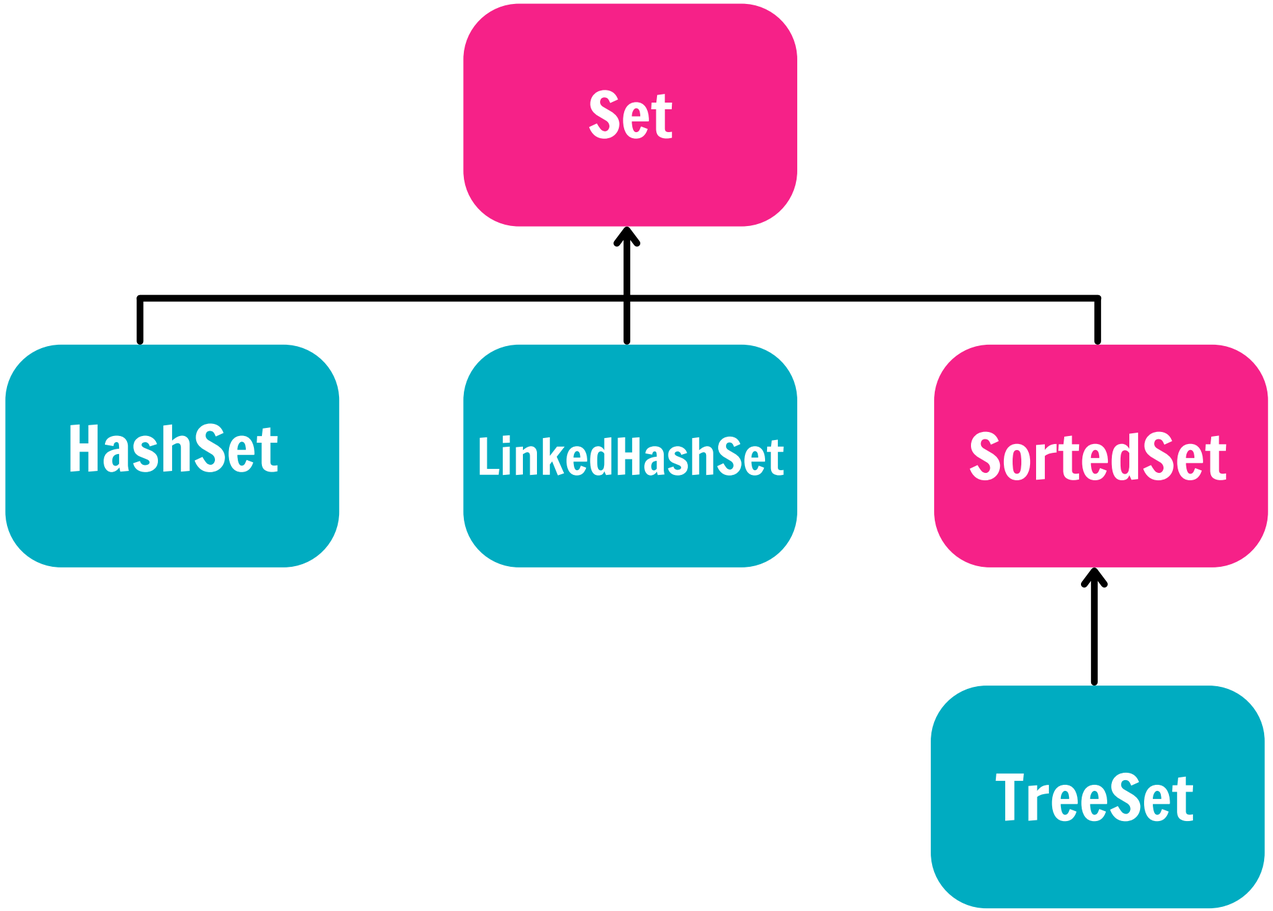


JAVA

Queue Interface（FIFO）

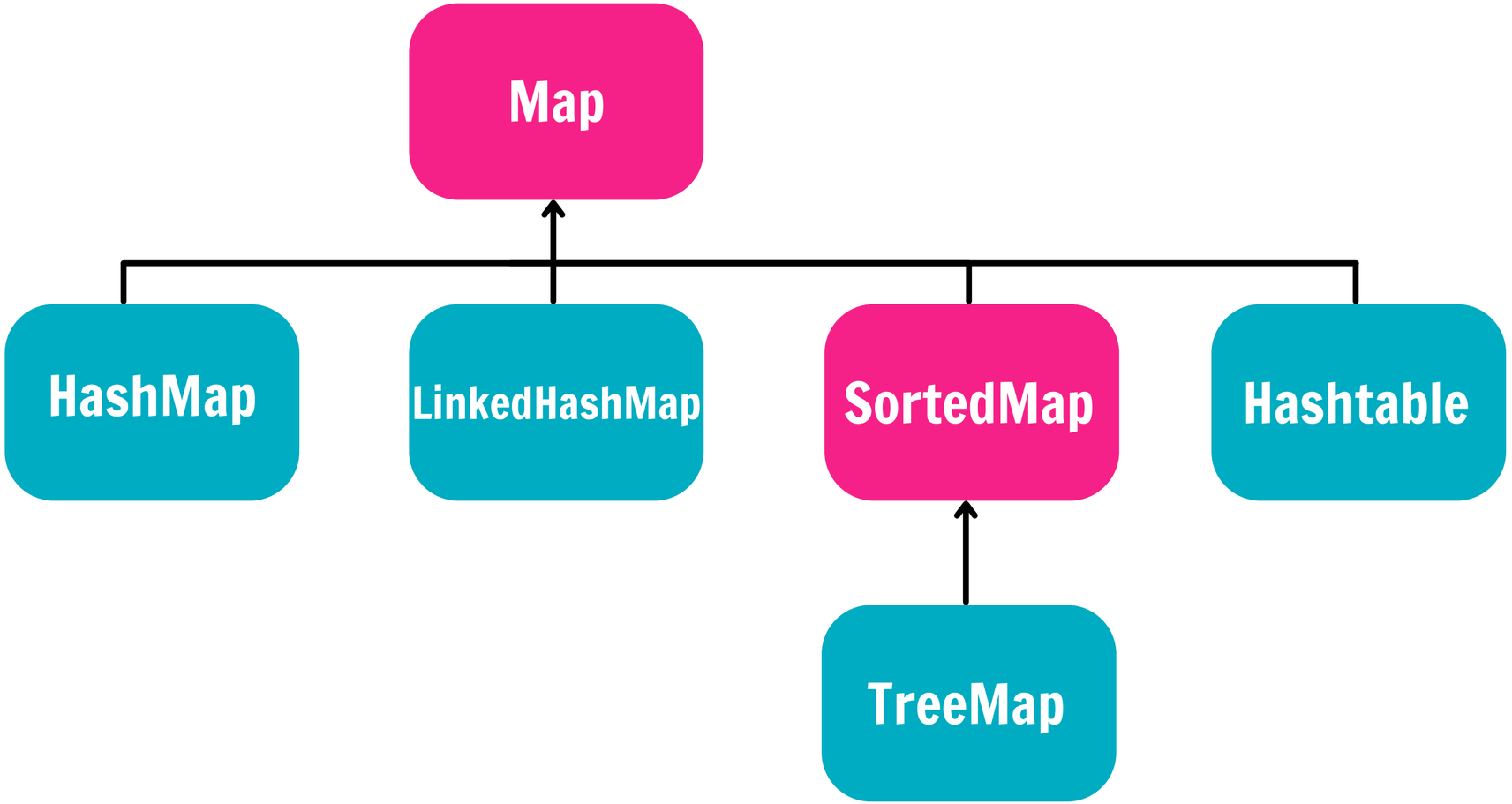


Set Interface



JAVA

Map Interface



Introduction to Linked List
Java

Linked List

LinkedList class Implementation (Collection Framework)

```
import java.util.*;

class LL {
   public static void main(String args[]) {
       LinkedList<String> list = new LinkedList<String>();
       list.add("is");
       list.add("a");
       list.addLast("list");
       list.addFirst("this");
       list.add(3, "linked");
       System.out.println(list);

       System.out.println(list.get(0));
       System.out.println(list.size());
       list.remove(3);
       list.removeFirst();
       list.removeLast();

       System.out.println(list);
   }
}
```

Apna College

Scratch Implementation (Important for BEGINNERS)

```
class LL {


   Node head;
   private int size;


   LL () {
       size = 0;
   }


   public class Node {
       String data;
       Node next;


       Node(String data) {
           this.data = data;
           this.next = null;
           size++;
       }
   }


   public void addFirst(String data) {
       Node newNode = new Node(data);
       newNode.next = head;
       head = newNode;
   }
```

Apna College

```java
public void addLast(String data) {
    Node newNode = new Node(data);

    if(head == null) {
        head = newNode;
        return;
    }

    Node lastNode = head;
    while(lastNode.next != null) {
        lastNode = lastNode.next;
    }

    lastNode.next = newNode;
}

public void printList() {
    Node currNode = head;

    while(currNode != null) {
        System.out.print(currNode.data+" -> ");
        currNode = currNode.next;
    }

    System.out.println("null");
```

Apna College

```
    }

    public void removeFirst() {
        if(head == null) {
            System.out.println("Empty List, nothing to delete");
            return;
        }

        head = this.head.next;
        size--;
    }

    public void removeLast() {
        if(head == null) {
            System.out.println("Empty List, nothing to delete");
            return;
        }

        size--;
        if(head.next == null) {
            head = null;
            return;
        }

        Node currNode = head;
        Node lastNode = head.next;
```

Apna College

```
        while(lastNode.next != null) {
            currNode = currNode.next;
            lastNode = lastNode.next;
        }

        currNode.next = null;
    }

    public int getSize() {
        return size;
    }




    public static void main(String args[]) {
        LL list = new LL();
        list.addLast("is");
        list.addLast("a");
        list.addLast("list");
        list.printList();

        list.addFirst("this");
        list.printList();
        System.out.println(list.getSize());

        list.removeFirst();
```

Apna College

```
        list.printList();



        list.removeLast();

        list.printList();

    }

}
```

How to insert in the middle of a Linked List (at a specified index ‘i’) ?

Scratch

```
public void addInMiddle(int index, String data) {

        if(index > size || index < 0) {

            System.out.println("Invalid Index value");

            return;

        }

        size++;



        Node newNode = new Node(data);

        if(head == null || index == 0) {

            newNode.next = head;

            head = newNode;

            return;

        }

         Node currNode = head;

        for(int i=1; i<size; i++) {

            if(i == index) {
```

Apna College

```
                Node nextNode = currNode.next;

                currNode.next = newNode;

                newNode.next = nextNode;

                break;

            }

            currNode = currNode.next;

        }

    }
```

LinkedList class

```
import java.util.*;



class LL {

    public static void main(String args[]) {

        LinkedList<String> list = new LinkedList<String>();



        list.addFirst("shradha");

        list.addFirst("name");

        list.addFirst("my");

        System.out.println(list);



        list.add(2, "is");

        System.out.println(list);

    }

}
```

Apna College

Homework Problems

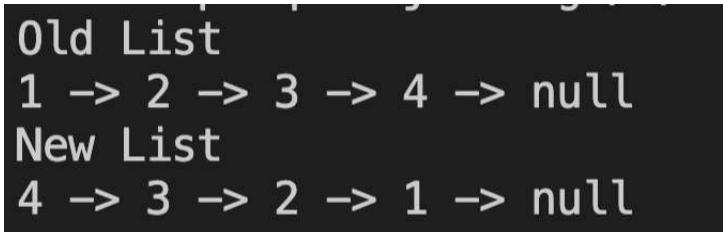
1. Make a Linked List & add the following elements to it : (1, 5, 7, 3 , 8, 2, 3). Search for the number 7 & display its index.
2. Take elements(numbers in the range of 1-50) of a Linked List as input from the user. Delete all nodes which have values greater than 25.

Apna College

Reverse a Linked List

Java

Reverse a Linked List without using extra space.



Iterative Method

Time complexity - O(n)

Space complexity - O(1)

```java
public void reverseList() {
        if(head == null || head.next == null) {
            return;
        }

        Node prevNode = head;
        Node currNode = head.next;
        while(currNode != null) {
            Node nextNode = currNode.next;
            currNode.next = prevNode;
            prevNode = currNode;
            currNode = nextNode;
        }
        head.next = null;
        head = prevNode;
    }
```

Apna College

Recursive Method

Time complexity - O(n)

Space complexity - O(1)

```
public Node reverseListRecursive(Node head) {

       //empty node || last node or only one node

       if(head == null || head.next == null) {

           return head;

       }


       Node newHead = reverseListRecursive(head.next);


       head.next.next = head;

       head.next = null;

       return newHead;

   }
```

Collections Method

Time complexity - O(n)

Space complexity - O(1)

```
LinkedList<Integer> list2 = new LinkedList<>();

       list2.add(1);

       list2.add(2);

       Collections.reverse(list2);
```

Apna College

Homework Problems

1. https://leetcode.com/problems/swap-nodes-in-pairs/
2. https://leetcode.com/problems/remove-nth-node-from-end-of-list/
3. https://leetcode.com/problems/reverse-linked-list-ii/
4. https://leetcode.com/problems/remove-nth-node-from-end-of-list/

Apna College

BEST Linked List Questions
Java

1. **Find the nth node from the end & remove it.**

Time complexity – O(n)

Space complexity – O(1)

```java
public ListNode removeNthFromEnd(ListNode head, int n) {
    if(head.next == null) {
        return null;
    }


    int size = 0;
    ListNode temp = head;
    while(temp != null) {
        temp = temp.next;
        size++;
    }

    //removing SIZEth node from last i.e. head
    if(n == size) {
        return head.next;
    }

    //find previous node
    int ptf = size - n; // position to find
    ListNode prev = head; // previous node
    int cp = 1; // current position

    while(cp != ptf) {
        prev = prev.next;
        cp++;
    }

    prev.next = prev.next.next;
    return head;
```

Apna College

```
}
```

2. **Check if a Linked List is a palindrome**

Time complexity – O(n)

Space complexity – O(1)

```
public ListNode getMiddle(ListNode head) {
    ListNode fast = head;
    ListNode slow = head;
    while (fast.next != null && fast.next.next != null) {
        fast = fast.next.next;
        slow = slow.next;
    }
    return slow;
}

public ListNode reverse(ListNode head) {
    ListNode prev = null;
    ListNode curr = head;

    while (curr != null) {
        ListNode next = curr.next;
        curr.next = prev;
        prev = curr;
```

Apna College

```
        curr = next;
    }
    return prev;
}



public boolean isPalindrome(ListNode head) {
    if(head == null || head.next == null) {
        return true;
    }

    ListNode firstHalfEnd = getMiddle(head);
    ListNode secondHalfStart = reverse(firstHalfEnd.next);
    ListNode firstHalfStart = head;

    while(secondHalfStart != null) {
        if(secondHalfStart.val != firstHalfStart.val) {
            return false;
        }
        secondHalfStart = secondHalfStart.next;
        firstHalfStart = firstHalfStart.next;
    }

    return true;
}
```

Apna College

3. **Detecting Loop in a Linked List.**

Time complexity - O(n)

Space complexity - O(1)

```
public boolean hasCycle(ListNode head) {

   ListNode slow = head;

   ListNode fast = head;



   while(fast != null && fast.next != null) {

       slow = slow.next;

       fast = fast.next.next;



       if(fast == slow) {

           return true;

       }

   }



   return false;

}
```

Homework Problems

1. Removing Loops in a Linked List.

   (Please try on your own first. The answer will be updated soon!)

Apna College

Queue

- Queue using Array

```java
//queue using array
public class QueueB {
   static class Queue {
       static int arr[];
       static int size;
       static int rear;

       Queue(int size) {
           this.size = size;
           arr = new int[size];
           rear = -1;
       }

       public static boolean isEmpty() {
           return rear == -1;
       }

       public static boolean isFull() {
           return rear == size-1;
       }

       public static void add(int data) {
           if(isFull()) {
               System.out.println("Overflow");
               return;
           }

           arr[++rear] = data;
       }

       //O(n)
       public static int remove() {
           if(isEmpty()) {
               System.out.println("empty queue");
               return -1;
           }
           int front = arr[0];
           for(int i=0; i<rear; i++) {
               arr[i] = arr[i+1];
```

```
            }
             rear-;
            return front;
        }

        public static int peek() {
            if(isEmpty()) {
                System.out.println("empty queue");
                return -1;
            }

            return arr[0];
        }
    }
    public static void main(String args[]) {
        Queue q = new Queue(5);
        q.add(1);
        q.add(2);
        q.add(3);
        System.out.println(q.remove());
        System.out.println(q.peek());
    }
}
```

Circular queue using array

```
//circular queue using array
public class QueueB {
    static class Queue {
        static int arr[];
        static int size;
        static int front = -1;
        static int rear = -1;

        Queue(int size) {
            this.size = size;
            arr = new int[size];
        }

        public static boolean isEmpty() {
            return rear == -1 && front == -1;
        }
```

```
public static boolean isFull() {
    return (rear+1)%size == front;
}

public static void add(int data) {
    if(isFull()) {
        System.out.println("Overflow");
        return;
    }
    //if it's the 1st element
    if(front == -1) {
        front = 0;
    }

    rear = (rear + 1)%size;
    arr[rear] = data;
}

public static int remove() {
    if(isEmpty()) {
        System.out.println("empty queue");
        return -1;
    }
    int res = arr[front];

    //if only 1 element is present
    if(front == rear) {
        front = rear = -1;
    } else {
        front = (front+1)%size;
    }

    return res;
}

public static int peek() {
    if(isEmpty()) {
        System.out.println("empty queue");
        return -1;
    }
```

```
            return arr[front];
        }
    }
    public static void main(String args[]) {
        Queue q = new Queue(5);
        q.add(1);
        q.add(2);
        q.add(3);
        q.add(4);
        q.add(5);
        System.out.println(q.remove());
        q.add(6);
        System.out.println(q.remove());
        q.add(7);

        while(!q.isEmpty()) {
            System.out.println(q.remove());
        }
    }
}
```

- Queue using Linked List

```
//queue using Linked List
public class QueueB {
    static class Node {
        int data;
        Node next;
        Node(int data) {
            this.data = data;
            next = null;
        }
    }

    static class Queue {
        static Node head = null;
        static Node tail = null;

        public static boolean isEmpty() {
            return head == null && tail == null;
        }
```

```
    public static void add(int data) {
        Node newNode = new Node(data);
        if(isEmpty()) {
            tail = head = newNode;
        } else {
            tail.next = newNode;
            tail = newNode;
        }
    }

    public static int remove() {
        if(isEmpty()) {
            System.out.println("empty queue");
            return -1;
        }
        int front = head.data;
        //single node
        if(head == tail) {
            tail = null;
        }
        head = head.next;
        return front;
    }

    public static int peek() {
        if(isEmpty()) {
            System.out.println("empty queue");
            return -1;
        }

        return head.data;
    }
}
public static void main(String args[]) {
    Queue q = new Queue();
    q.add(1);
    q.add(2);
    q.add(3);
    q.add(4);
    q.add(5);
```

```
        while(!q.isEmpty()) {
            System.out.println(q.peek());
            q.remove();
        }
    }
}
```

- Java Collection Framework

```
//queue using Java Collection Framework
import java.util.*;

public class QueueB {
    public static void main(String args[]) {
        //Queue<Integer> q = new LinkedList();
        Queue<Integer> q = new ArrayDeque();
        q.add(1);
        q.add(2);
        q.add(3);
        q.add(4);
        q.add(5);

        while(!q.isEmpty()) {
            System.out.println(q.peek());
            q.remove();
        }
    }
}
```

- Queue using 2 stacks

```
//queue using 2 stacks
import java.util.*;

public class QueueB {
    static class Queue {
```

```
        static Stack<Integer> s1 = new Stack<>();
        static Stack<Integer> s2 = new Stack<>();

        public static boolean isEmpty() {
            return s1.isEmpty();
        }

        public static void add(int data) {
            while(!s1.isEmpty()) {
                s2.push(s1.pop());
            }
            s1.push(data);
            while(!s2.isEmpty()) {
                s1.push(s2.pop());
            }
        }

        public static int remove() {
            return s1.pop();
        }

        public static int peek() {
            return s1.peek();
        }
    }
    public static void main(String args[]) {
        Queue q = new Queue();
        q.add(1);
        q.add(2);
        q.add(3);

        while(!q.isEmpty()) {
            System.out.println(q.peek());
            q.remove();
        }
    }
}
```

Java - DSA

Trees

1. Build Tree from given Preorder Sequence

```java
//Build a Tree from its Preorder traversal

public class BinaryTreesYT {
   static class Node {
       int data;
       Node left;
       Node right;

       Node(int data) {
           this.data = data;
           this.left = null;
           this.right = null;
       }
   }

   static class BinaryTree {
       static int idx = -1;
       public static Node buildTree(int nodes[]) {
           idx++;
           if(nodes[idx] == -1) {
               return null;
           }
           Node newNode = new Node(nodes[idx]);
           newNode.left = buildTree(nodes);
           newNode.right = buildTree(nodes);
           return newNode;
       }
   }

   public static void main(String args[]) {
       int nodes[] = {1, 2, 4, -1, -1, 5, -1, -1, 3, -1, 6, -1, -1};
       BinaryTree tree = new BinaryTree();

       Node root = tree.buildTree(nodes);
       System.out.println(root.data);
   }
```

```
}
```

2. Tree Traversals

a. Preorder

```
public static void preorder(Node root) {
        if(root == null) {
            System.out.print(-1+" ");
            return;
        }
        System.out.print(root.data+" ");
        preorder(root.left);
        preorder(root.right);
    }
```

b. Inorder

```
public static void inorder(Node root) {
        if(root == null) {
            System.out.print(-1+" ");
            return;
        }
        inorder(root.left);
        System.out.print(root.data+" ");
        inorder(root.right);
    }
```

c. Postorder

```
public static void postorder(Node root) {
        if(root == null) {
            System.out.print(-1+" ");
            return;
        }
        postorder(root.left);
        postorder(root.right);
        System.out.print(root.data+" ");
    }
```

d. Level Order

```
public static void levelOrder(Node root) {
        if(root == null) {
            return;
        }
        Queue<Node> q = new LinkedList<>();
        q.add(root);
        q.add(null);
        while(!q.isEmpty()) {
            Node curr = q.remove();
            if(curr == null) {
                System.out.println();
                //queue empty
                if(q.isEmpty()) {
                    break;
                } else {
                    q.add(null);
                }
            } else {
                System.out.print(curr.data+" ");
                if(curr.left != null) {
                    q.add(curr.left);
                }
                if(curr.right != null) {
                    q.add(curr.right);
                }
            }
        }
    }
```

3. Height of Tree

```
public static int height(Node root) {
        if(root == null) {
            return 0;
        }

        int leftHeight = height(root.left);
        int rightHeight = height(root.right);
        return Math.max(leftHeight, rightHeight) + 1;
```

```
    }
```

4. Count of Nodes of Tree

```
public static int countOfNodes(Node root) {
        if(root == null) {
            return 0;
        }

        int leftNodes = countOfNodes(root.left);
        int rightNodes = countOfNodes(root.right);
        return leftNodes + rightNodes + 1;
    }
```

5. Sum of Nodes of Tree

```
public static int sumOfNodes(Node root) {
        if(root == null) {
            return 0;
        }

        int leftSum = sumOfNodes(root.left);
        int rightSum = sumOfNodes(root.right);
        return leftSum + rightSum + root.data;
    }
```

6. Diameter of Tree - Approach1 O(N^2)

```
public static int diameter(Node root) {
        if(root == null) {
            return 0;
        }

        int diam1 = height(root.left) + height(root.right) + 1;
        int diam2 = diameter(root.left);
        int diam3 = diameter(root.right);

        return Math.max(diam1, Math.max(diam2, diam3));
    }
```

7. Diameter of Tree - Approach2 O(N)

```
public static TreeInfo diameter(Node root) {
       if(root == null) {
           return new TreeInfo(0, 0);
       }

       TreeInfo leftTI = diameter(root.left);
       TreeInfo rightTI = diameter(root.right);

       int myHeight = Math.max(leftTI.height, rightTI.height) + 1;

       int diam1 = leftTI.height + rightTI.height + 1;
       int diam2 = leftTI.diam;
       int diam3 = rightTI.diam;

       int myDiam = Math.max(diam1, Math.max(diam2, diam3));

       return new TreeInfo(myHeight, myDiam);
   }
```

8. Subtree of another tree

```
 public boolean isIdentical(TreeNode root,TreeNode subRoot){
       if(subRoot == null && root == null){
           return true;
       }
       if(root == null || subRoot == null){
           return false;
       }
       if(root.val == subRoot.val){
           return isIdentical(root.left, subRoot.left) &&
isIdentical(root.right, subRoot.right);
       }
       return false;
   }

   public boolean isSubtree(TreeNode root, TreeNode subRoot) {
       if(subRoot == null){
           return true;
       }
       if(root == null){
           return false;
```

```
        }
        if(isIdentical(root, subRoot)){
            return true;
        }
        return isSubtree(root.left, subRoot) || isSubtree(root.right, subRoot);
    }
```

HashSet in Java

```java
import java.util.HashSet;
import java.util.Iterator;

public class Hashing {
   public static void main(String args[]) {
       HashSet<Integer> set = new HashSet<>();

       //Add
       set.add(1);
       set.add(2);
       set.add(3);
       set.add(1);

       //Size
       System.out.println("size of set is : " + set.size());

       //Search
       if(set.contains(1)) {
           System.out.println("present");
       }

       if(!set.contains(6)) {
           System.out.println("absent");
       }

       //Delete
       set.remove(1);
       if(!set.contains(1)) {
           System.out.println("absent");
       }

       //Print all elements
       System.out.println(set);

       //Iteration - HashSet does not have an order
       set.add(0);
       Iterator it = set.iterator();
        while (it.hasNext()) {
           System.out.print(it.next() + ", ");
       }
```

```
        System.out.println();

        //isEmpty
        if(!set.isEmpty()) {
            System.out.println("set is not empty");
        }
    }
}
```

HashMap in Java

```java
import java.util.*;

public class Hashing {
   public static void main(String args[]) {
       //Creation
       HashMap<String, Integer> map = new HashMap<>();

       //Insertion
       map.put("India", 120);
       map.put("US", 30);
       map.put("China", 150);

       System.out.println(map);

       map.put("China", 180);
       System.out.println(map);


       //Searching
       if(map.containsKey("Indonesia")) {
           System.out.println("key is present in the map");
       } else {
           System.out.println("key is not present in the map");
       }

       System.out.println(map.get("China")); //key exists
       System.out.println(map.get("Indonesia")); //key doesn't exist

       //Iteration (1)
       for( Map.Entry<String, Integer> e : map.entrySet()) {
           System.out.println(e.getKey());
           System.out.println(e.getValue());
       }

       //Iteration (2)
       Set<String> keys = map.keySet();
       for(String key : keys) {
           System.out.println(key+ " " + map.get(key));
       }
```

```
        //Removing
        map.remove("China");
        System.out.println(map);


    }
}
```