Date 12/8/2020

Types of Statements:

There are 3 types of statements:

1. Linear/Sequential statements: which are executed one after other
2. Conditional: which are executed on the basis of a condition
3. Loop: which are executed more than once.

Conditional statements:

There are two types of conditional statements:

1. If statement
2. Switch statement

If statement:

There are four forms of if:

1. if
2. If – else
3. If –else if—else(Ladder if)
4. If within if(Nested if)
5. Simple if:

Syn: if (condition)

{

//statements

}

For example:

Int N = 56;

If (N > 20)

{

Sop(“No. is greater than 20”);

}

1. If else

Syn: if (condition)

{

//statements

}

else

{

//statements

}

For example:

Int N = 56;

If (N > 20)

{

Sop(“No. is greater than 20”);

}

else

{

Sop(“No…”);

}

13/8/2021

Loops:

Hello

Hello – 100times

Types of Loops:

1. Entry controlled
2. Exit controlled

Entry controlled: are the loops which are checked first, if the condition is true only then the statements will be executed otherwise not.

1. For loop
2. While loop

Exit controlled: are the loops which are checked at the time of exit

1. Do---while loop

Things to look after:

1-100

1. Initialization
2. Condition
3. Modification

1 – 100 , 100 - 1

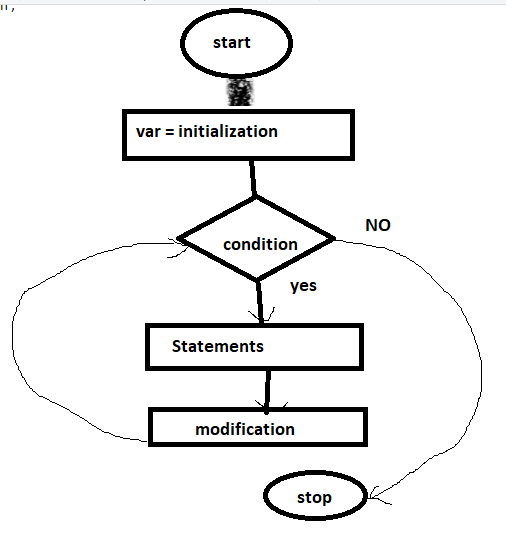
1. For loop:

Syntax: for (var = initialization ; condition ; modification)

{

//statements

}

I = i+1 or i+=1

For (i = 1 ; i<=100 ; i++) i= 2, 3,,,,,,,100,101

{

Sop(i);

}

1

2

3

-

- 99

100

I++ , ++i

B = ++i

C = i++

While loop:

Syn :

Var = initialization;

While(condition)

{

//statements;

Modification;

}

Date 16/8/21:

1. WAP to input ‘n’ terms from the user and display its Fibonacci series

N = 7

0,1 ,1,2,3,5,8

1. WAP to input ‘n’ terms from the user and display Lucas series

N = 5

0,1,2,5,10

0+1+1

1+2+2

2+5+3

5+10+4

1. WAP to input ‘n’ terms and a value for x from the user:

Val = 2, n=4

22/2! + 23/3!+24/4!+25/5!

Operations:

1. Power
2. Factorial
3. Divide
4. Add

**While loop:**

Syn :

Var = initialization;

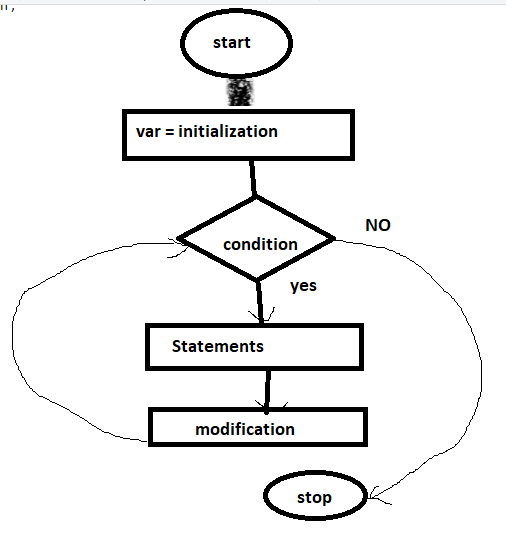
While(condition)

{

//statements;

Modification;

}



For example: display numbers from 1-100

WAP to input any number and display sum of digits

N = 2345 , r, sum =0;

Step 1:

Num = 2345 >0

R = 2345 % 10 🡺 5

Sum = sum + r; 🡺 0+5 🡺5

Num = 2345 /10 🡺234

Num = 234

Step 2: 234 > 0

R = 234 % 10 🡺 4

Sum = sum + r 🡺5+4 🡺9

Num = num /10 🡺 234 /10 🡺23

Step 3: Num = 23 > 0

R = 23 % 10 🡺 3

Sum = 9+3🡺12

Num = num/10 🡺23/10🡺2

Step 4: num = 4>0

R = 4%10 = 4

Sum = sum + r🡺12+4🡺16

Num = num/10 🡺4/10 🡺0

Num = 0

2+3+4+5 = 14

1234 , 6789

1. WAP any 4 digit number and reverse it.

Num = 123 🡺 “Invalid number”

Num = 12345 🡺”Invalid number”

Num = 2345

Output = 5432

1. WAP to any number and check if it is Armstrong or not

Num = 1234

Digits = 4

1^4+2^4+3^4+4^4

1+16+81+256 = 354

Not an Armstrong number

153🡺armstrong number

Do –while loop

Syn:

var = initialization;

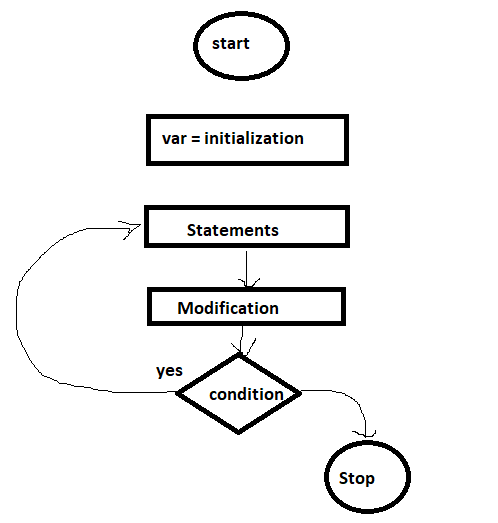
do

{

//Statements;

Modification;

}while(condition);



NOTE: Exit controlled loops are executed at least once.

WAP to display numbers from 1-10;

1. WAP to input any number and display its factorial using do—while.

Date 20/8/2021

Input

Display

Sorting

Searching

Searching: to find an element from set of elements

Complexity:

Best case:

Average case

Worst case

Time complexity:

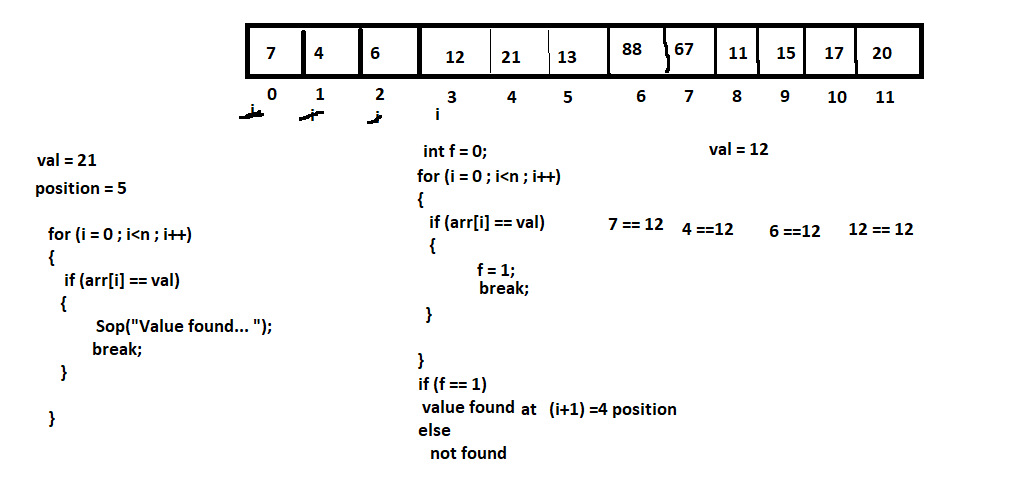
Space:

There are two types of searching :

1. Linear / Sequential Search
2. Binary Search

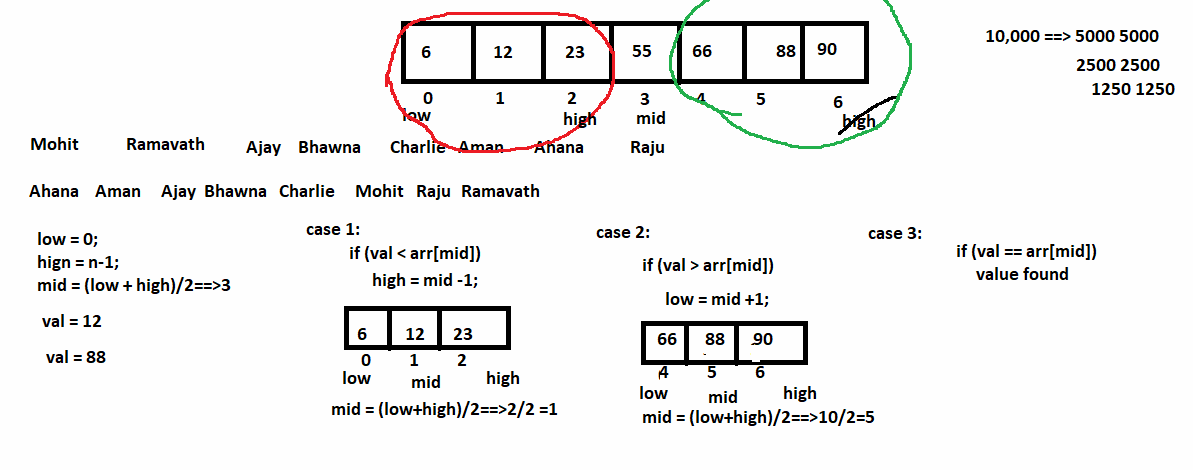
Linear / Sequential Search: Sorting is not required in this search.

It is good in case if you have lesser numbers of elements



Binary Search: Sorting is must.

Best when you have larger number of elements.



Date 21/8/2021

int arr[] = {2,34,5,6,7};

for (i = 0 ; i<arr.length ; i++)

{ arr[i] = sc.nextInt();

Sop(arr[i]); ==>2 34 5 6 7

}

For each/Enchanced for loop

1. used for accessing the elements

2. no need to give initialization,condition and modification

for (int i :arr)

{

Sop(i); ==>2 34 5 6 7

}

Functions: dividing the code into small independent modules

Main()

{

Int A b c

Enter three numbers….

Sum = a+b+c;

Sop(sum);

Dafs

Dddd

Ddd

Enter three numbers….

Sum = a+b+c;

Sop(sum);

Aa

Aa

Aa

Enter three numbers….

Sum = a+b+c;

Sop(sum);

}

Println()

Categories of Functions/Methods:

1. Predefined: println(), nextInt(),……
2. User defined

Categories of user defined methods:

1. No return value, no argument
2. No return value, with argument
3. Return value, no argument
4. Return value, with argument

Method Segments:

1. Method declaration
2. Method definition
3. Method calling

Method declaration/ Method prototype: used in abstract classes and interfaces

Syntax:

Return\_type method\_name(data\_type arg1 , data\_type arg2,….);

1. Void sum();
2. Void sum(int a, int b);
3. Int sum( );
4. Int sum(int a , int b);

Method Definition: Set of statements/ definition will be given

Syntax: Return\_type method\_name(data\_type arg1 , data\_type arg2,….)

{

//defintion

}

For example: void sum()

{

//statements }

Method calling:

Syntax: value = method\_name(arg\_list);

Programs using 2nd category:

1. WAP to input any number and display its factorial
2. WAP to input ‘n’ terms and display Fibonacci series

Int A[] = {2,3,4,5,6,7}

Variable: is a name given to a memory location. A variable can have a single value at a time.

A

5

A12AA

Arrays: it is a collection of elements of similar type. It occupies continuous memory location.

It is also called as subscripted variable.

Type of Arrays:

1. Single Dimensional
2. Multi Dimensional

Single (1-D) :

Syntax :

1. Array Declaration:

Data\_type array\_name[];

Or

Data\_type [] array\_name;

For example: int arr[];

1. Memory Allocation:

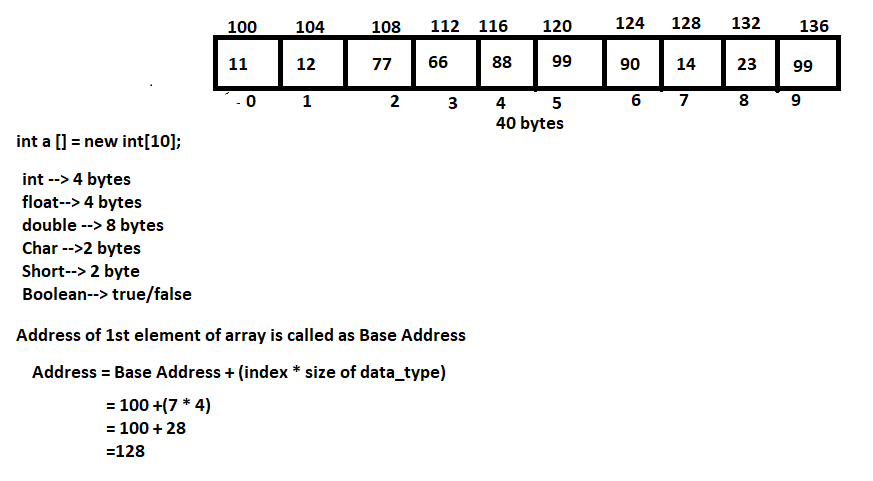
Array\_name = new data\_type[size];

For example: arr = new int[10];

Combine 1 and 2

Int arr[] = new int [10];

Memory Representation:



2-D Array: Array of Arrays. Used to represent data in a tabular/matrix format.

Syn:

1. Array Declaration:

data\_type array\_name[][];

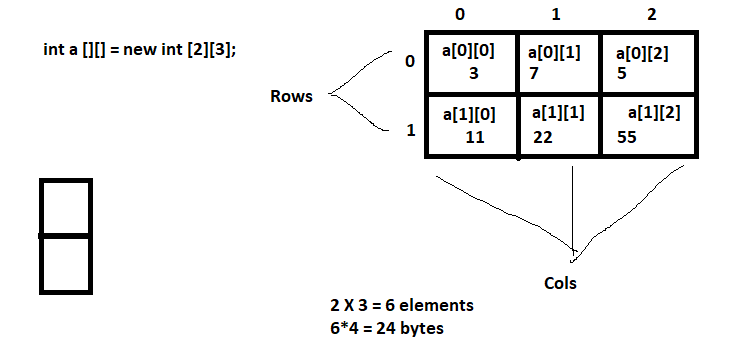
For example: int arr[][];

1. Memory Allocation: array\_name = new data\_type[row\_size][col\_size];

Arr = new int [2][3];

Combine 1 and 2

Int arr [][] = new int [2][3];



Date 23/8/2020:

String : Object which represents sequence of characters.

It is a final class

**Immutable: cannot be changed**

**String name =** “Ahana”;

**StringBuilder** and **StringBuffer** are mutable classes. These are same. StringBuilder is faster than StringBuffer.

Name: aradhana

Password:Asb123#$%%^

Phon:234567811

Email:aradhana@niit.com

Regular Expressions(RegEx):

[]🡺subscripts: used in 1D array

[]🡺Character class

Pattern , Matcher 🡺 java.util.regex;

[0-9][0-9][][][][][][][] 🡺2 digit

[^0-9]

Predefined Character Class:

\d: digit

\D: not a digit

Date 26/8/2021:

Abstraction: is a process of hiding background details and showing required output.

Encapsulation: wrapping up of data members/variables/attributes/states and methods/behaviour/functions into a single unit (Class).

Class: it is blueprint or template from which objects are created.

Object: variable of class/ instance of class/ runtime entity

Class HumanBeing

{

2 eyes, 2 legs , 1 tongue…..

Left\_eye , right\_eye, left\_leg , right\_leg

Drink();

Eat(), walk(), sing()….

}

HumanBeing Ramavath, Shashank, mohit //objects

Int k;

Class Student

{

Sid, name, course, fee, address, contact,emailId //data member/attributes/variable

Study() //method/function/behaviour

}

Syntax for creating class:

class Class\_Name

{

Access\_specifier data\_type var\_name; data\_members/ attributes/instance variable/states

-

Access\_specifier return\_type method\_name(arg1, arg2,…..) members

{ behaviors/methods

//definition

}

-

}

Here, class is a keyword which is used to define a class.

Class\_Name is valid name of the class.

Access specifier specifies the accessibility of the members of the class. We have 4 access specifiers in Java:

Public, protected, private, default (by default)

Syntax to create object:

Class\_Name obj\_name = new Class\_name();

Here, Class\_Name is name of the class for which object is to be created

Obj\_name is a valid identifier

new: is a keyword used to allocate memory dynamically (memory will be created in Heap Memory)

Class\_Name(): is constructor

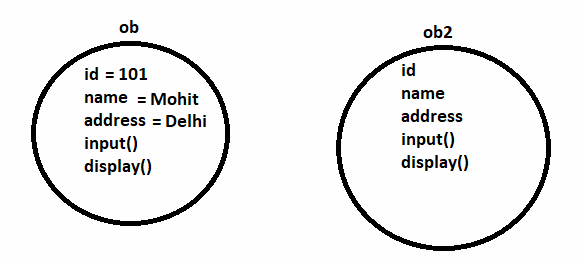
Scanner sc=new Scanner(System.in);

WAP to input information of a Student (id, name, address) and display.

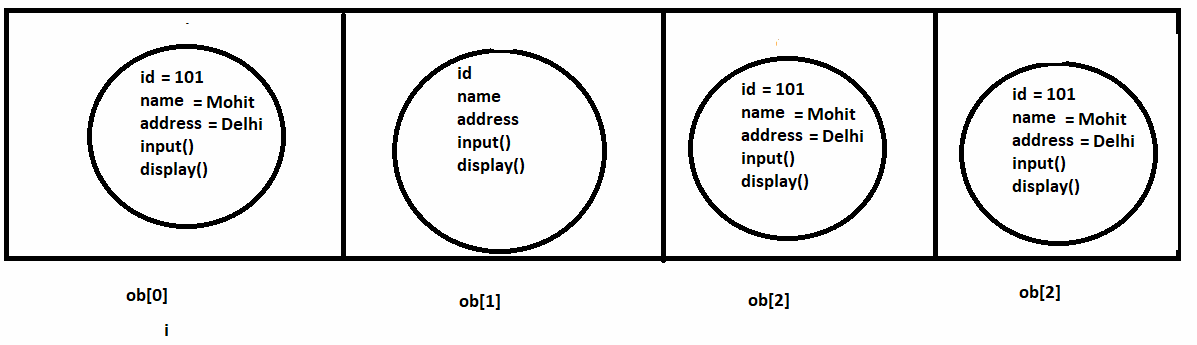
ClassName: Student

Attributes: id, name and Address

Methods/behaviors: input(), display()



Array of Objects:



WAP to input information of ‘n’ Employees (id, name, dept, salary and address) and perform following:

1. Display all employees
2. Search for employees getting salary <=10000
3. Increase the salary by 2000 for those who are getting <=10000

1 A IT 8000 SouthEx

2 B HR 12000 Mumbai

3 A Accounts 5000 SouthEx

1. Sort the objects on the basis of Employee Id

Constructors:

1. These are special methods which are used to create Objects and to initialize the Object (initialize the data members)
2. These are special methods because they have the same name as the name of the class
3. They do not have any return type/value not even void
4. These are called automatically at the time of creating objects.
5. These should be defined as public method
6. When a class defines more than one constructor then this is called Constructor Overloading.
7. There are three types of constructors:

* Default constructor
* Parametrized constructor
* Copy Constructor

**Default: if we do not define any constructor in the class, then default constructor is called by the compiler, so that objects can be created.**

**This type of constructor is used to initialize the object/data members with some default values.**

Syntax: public Class\_Name()

{

Data\_member = value;

-

}

**Parameterized: this type of constructor takes primitive type of parameters to initialize the data members/object**

Syntax:

Public Class\_Name(data\_type param1, data\_type param2,,…)

{

Data\_member1 = param1;

Data\_member2= param2;

}

**Copy: this type of constructor is used to copy one object to another. It takes an object as argument.**

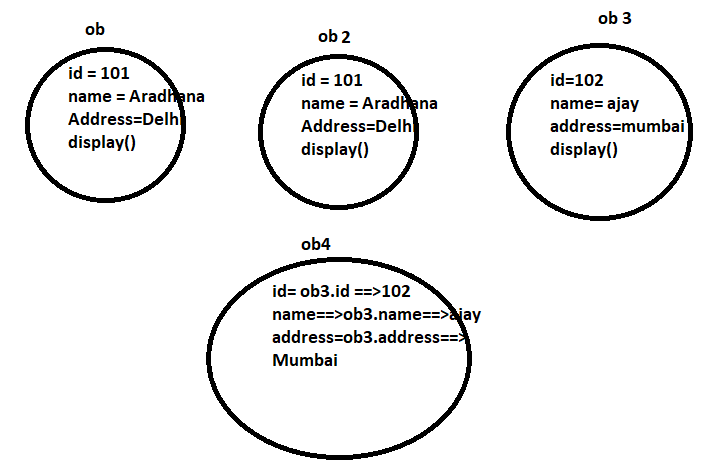
Syntax: public Class\_Name(Class\_Name ObjName)

{

Data\_member = ObjName.data\_member;

-

}



Date 27/8/2021

Static modifier

Static can be used with members of the class(data member and method).

A can also be a static (nested class)

Static variable: a static variable can be defined using static keyword.

If we do not initialize any static variable then it is automatically initialized. In case of numeric value it is initialized by 0 and in case of String value it is initialized by null.

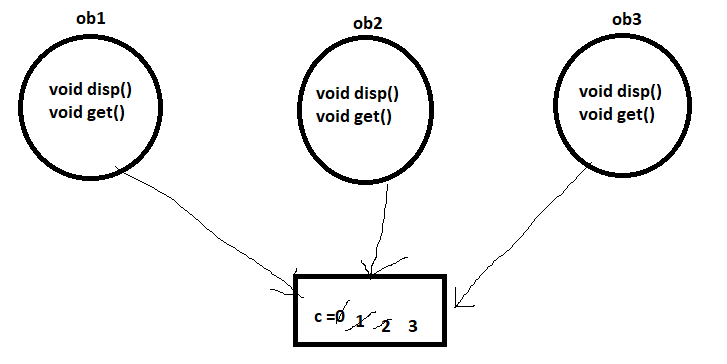
Static members are not associated with the object they are directly associated with the class.

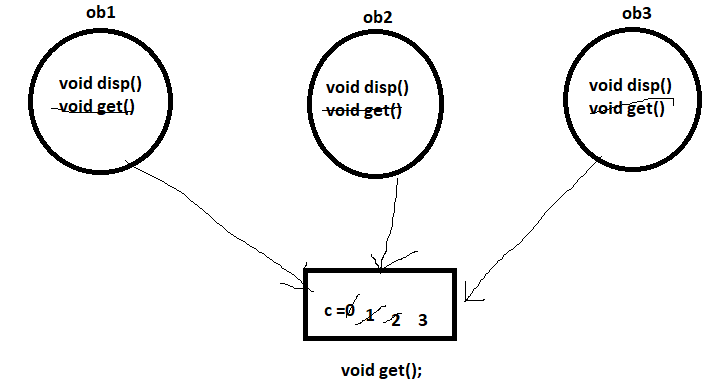
It means memory for the static members is not allocated in the object, it is allocated somewhere in the JVM.

Whenever we want to make members sharable then we can use static.

Static variables can be accessed in static as well as non-static methods

Static methods can access only static variables.





Final modifier is used for restrictions:

1. Variable: when it is used with variable, it will become constant means value cannot be changed during the execution of the program.

For example: final int MAX = 50;

MAX++; //ERROR

According to the Java naming conventions, constants should be defined in upper case.

1. Method: if it is used with method name, then that method cannot be overridden.

Class A

{

Public final void show(){ }

}

Class B extends A

{

Public void show() //ERROR🡪cannot override final method

{}

}

1. Class: if it is used with class name, then that class cannot be inherited or extended

For example:

Final Class A

{

}

Class B extends A //ERROR 🡪cannot extend final class.

{

}

WAP

Date 9/9/2021

Data Structure:

Linear and non Linear

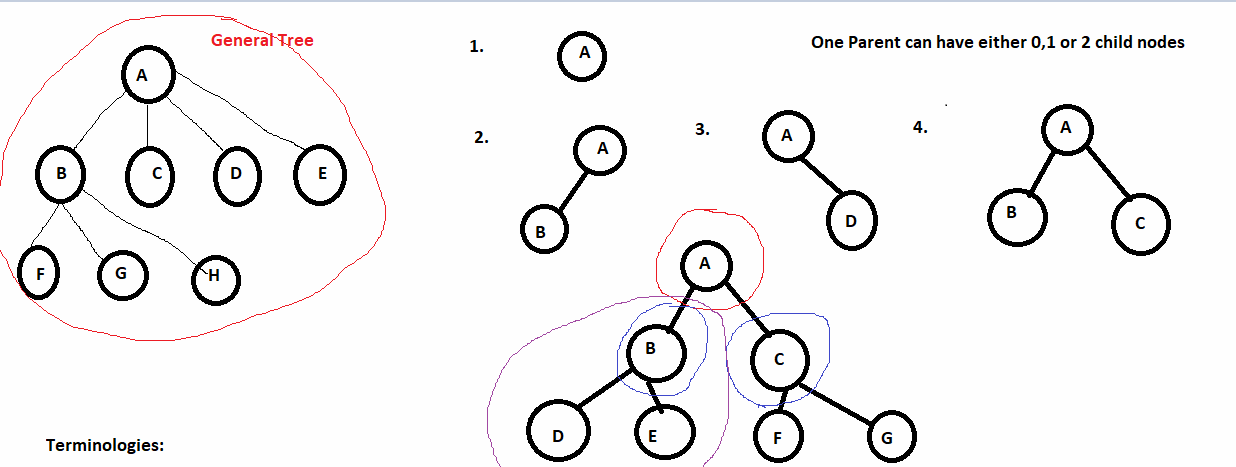
Linear DS: are those which are accessed sequentially

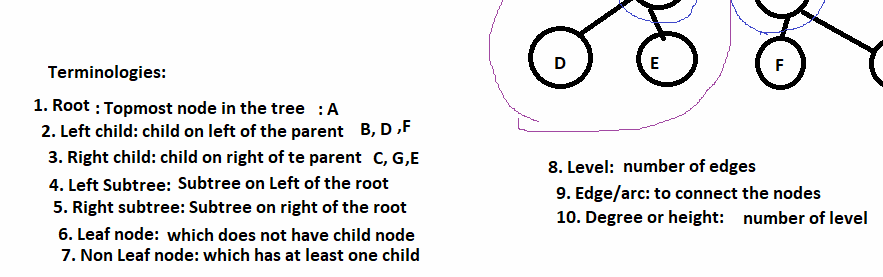
Arrays, Linked List, Stack and Queue

Non Linear DS: which cannot be traversed sequentially

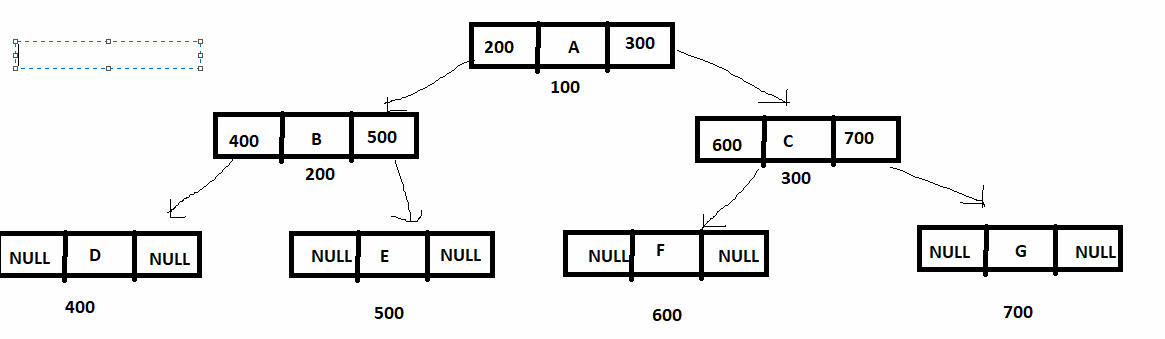
Trees and Graphs

Trees:





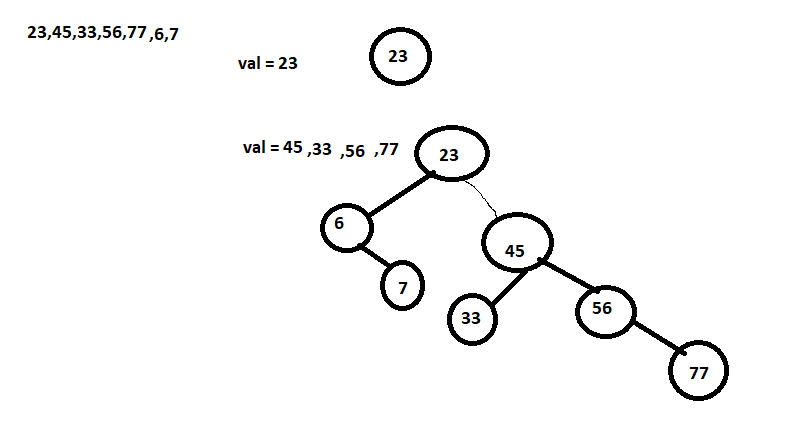
Linked List Representation of Binary Tree:



Binary Search Tree (BST):

1. Value which is lesser than the Parent will be put in the left side.
2. Value which is greater than the Parent will be put in the Right side.

Creation of BST:



Binary Tree Traversal Techniques:

1. Inorder
2. Preorder
3. Postorder

Inorder (Left Root Right):

1. Traverse the Left Subtree in Inorder
2. Visit the Root
3. Traverse the Right Subtree in Inorder

Preorder (Root Left Right):

1. Visit the Root
2. Traverse the Left subtree in Preorder
3. Traverse the Right subtree in Preorder

Postorder (Left Right Root):

1. Traverse the Left Subtree in Postorder
2. Traverse the Right Subtree in Postorder
3. Visit the Root

