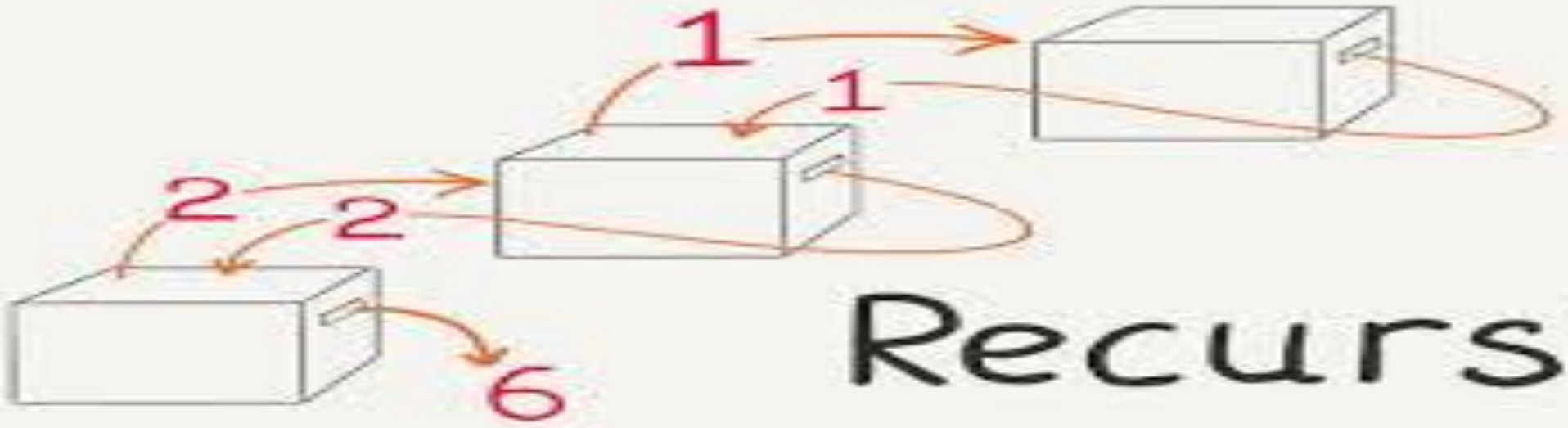
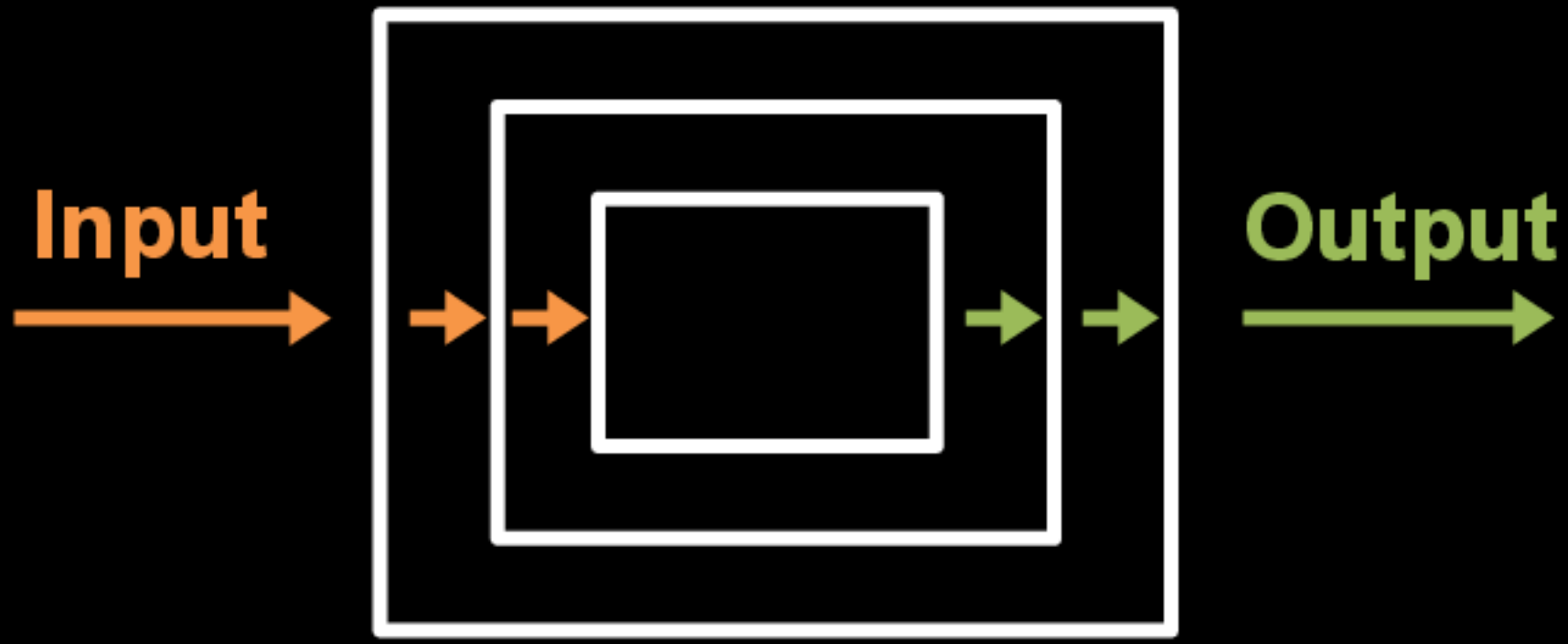


Topics

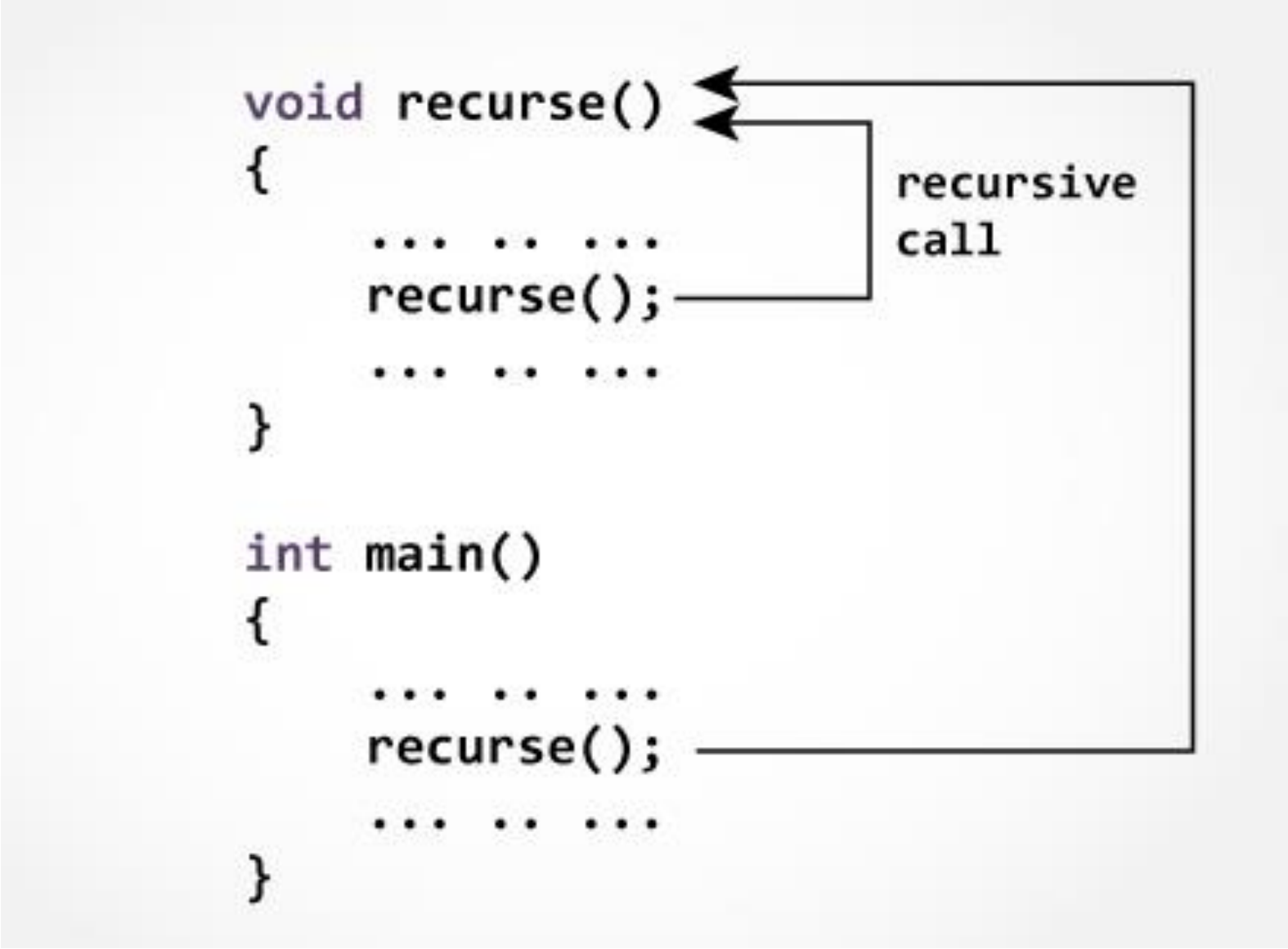
1. Recursive definitions and Processes
2. Writing Recursive Programs
3. Efficiency in Recursion
4. Towers of Hanoi problem.

Recursion



- **What is Recursion?**
  The process in which a function calls itself directly or indirectly is called **recursion** and the corresponding function is called as **recursive function**.
- Using recursive algorithm, certain problems can be solved quite easily.
- Examples of such problems are
- Towers of Hanoi (TOH),
- Inorder/Preorder/Postorder Tree Traversals,
- DFS of Graph, etc.

How does Recursion works?

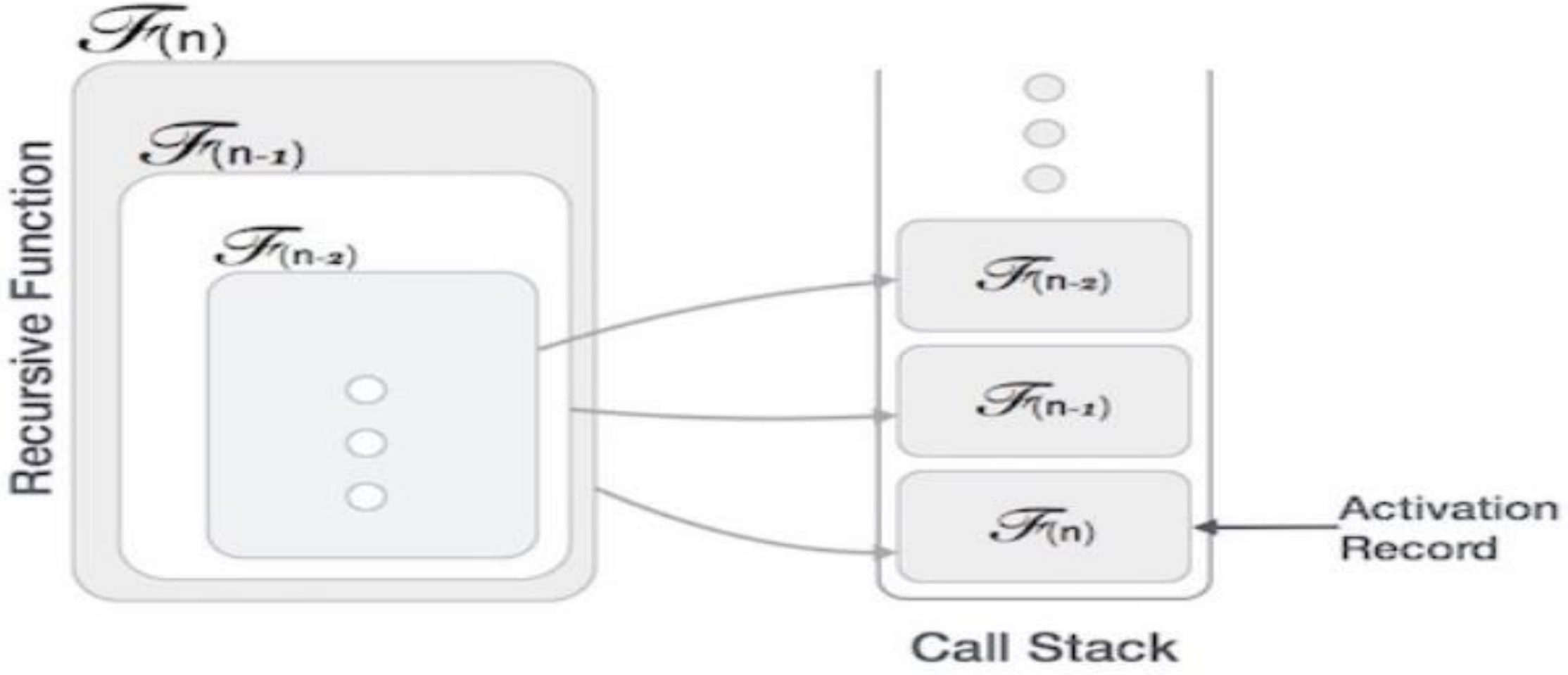


Recursion in Java

- **Recursion in java is a process in which a method calls itself continuously. A method in java that calls itself is called recursive method.**
- **It makes the code compact but complex to understand.**
- **Syntax:**

```
returntype methodname(){
//code to be executed
methodname( );//calling same method
}
```

How Data Structure Recursive function is implemented?



Outline of a Recursive Function

```
if (answer is known)
    provide the answer & exit
else
    call same function with
    a smaller version
    of the same problem
```

base case

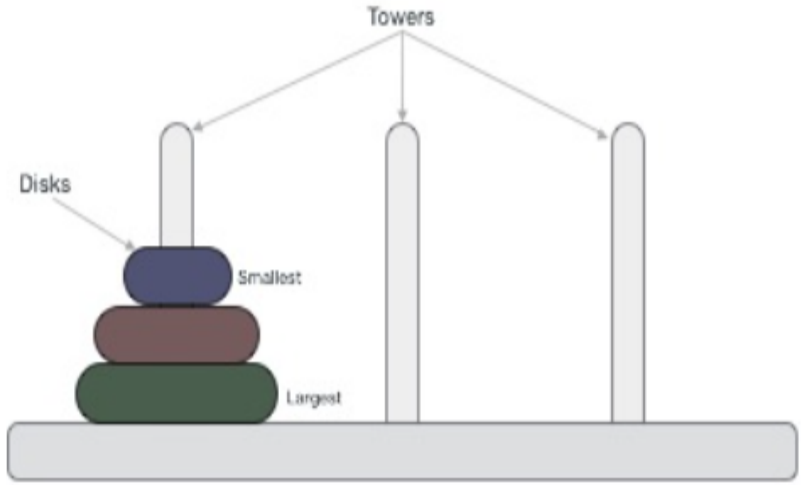
recursive case

What is Tower of Hanoi?

- A mathematical puzzle consisting of three towers and more than one ring is known as Tower of Hanoi.

- Tower of Hanoi

- The rings are of different sizes and are stacked in ascending order, i.e., the smaller one sits over the larger one. In some of the puzzles, the number of rings may increase, but the count of the tower remains the same.



- **In pseudocode, this looks like the following.**
- **At the top level, we'll call MoveTower with**
  - disk=5, source=A, dest=B, and spare=C.
- **FUNCTION MoveTower(disk, source, dest, spare):**

```
IF disk == 0, THEN:
    move disk from source to dest
ELSE:
    MoveTower(disk - 1, source, spare, dest)   // Step 1 above
    move disk from source to dest              // Step 2 above
    MoveTower(disk - 1, spare, dest, source)   // Step 3 above
END IF
```

Home Work

- Implement Tower of Hanoi Program
- No of Disk=3
- No of Disk=5
- No of Disk=n

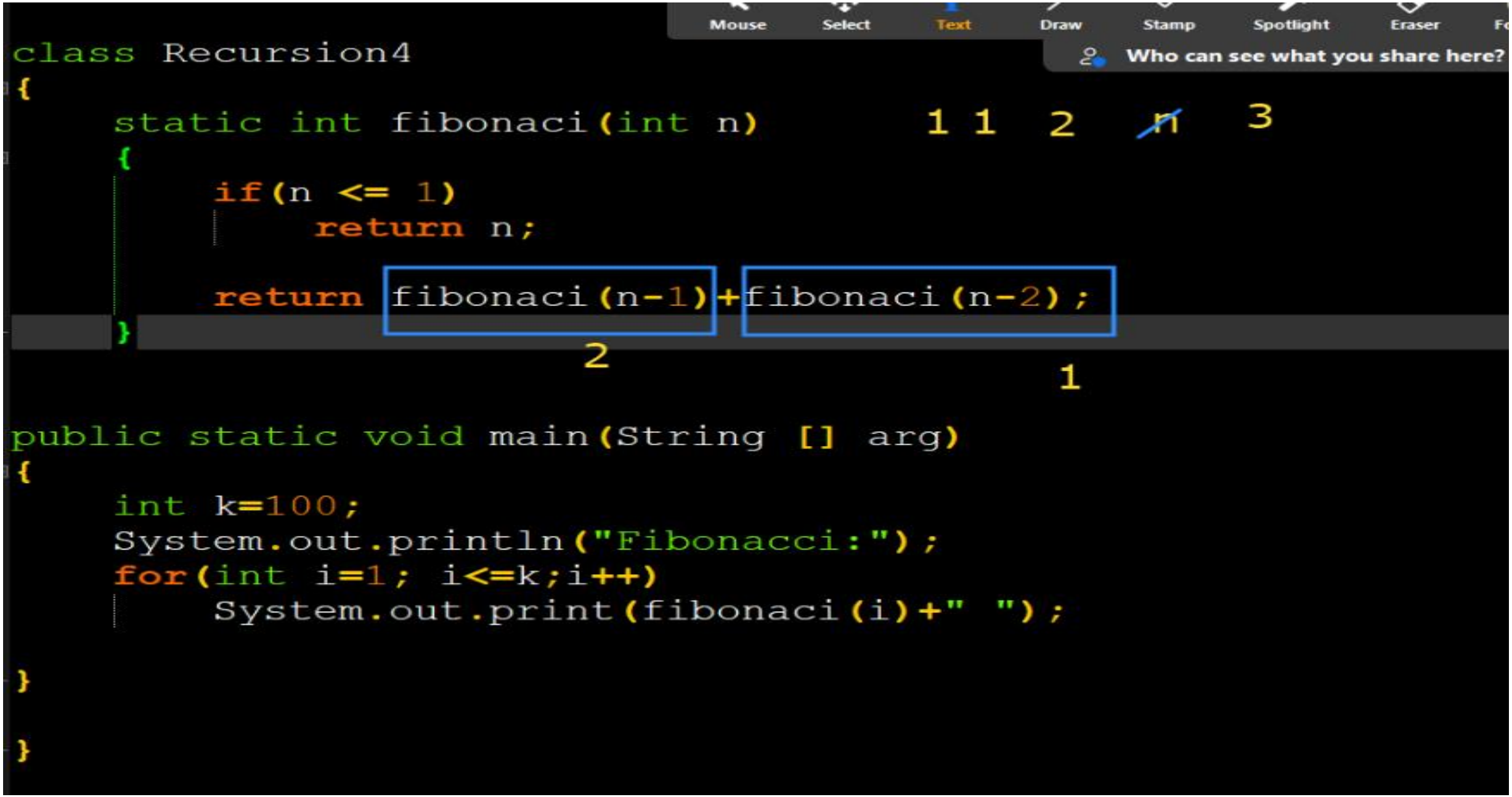
Assignment 1

1. Print a series of numbers with recursive Java methods
2. Sum a series of numbers with Java recursion
3. Calculate a factorial in Java with recursion
4. Print the Fibonacci series with Java and recursion
5. A recursive Java palindrome checker

Home Work

Why Algorithms?

- Fibonacci numbers
  - Compute first N Fibonacci numbers using iteration.
  - … using recursion.
- Write the code.
- Try for N=5, 10, 20, 50, 100
- What do you see? Why does this happen?


Mouse
Select
Text
Draw
Stamp
Spotlight
Eraser
Who can see what you share here?
class Recursion4
{
static int fibonaci(int n)
1 1 2 n 3
if(n <= 1)
return n;
return fibonaci(n-1)+fibonaci(n-2);
2
1
public static void main(String [] arg)
{
int k=100;
System.out.println("Fibonacci:");
for(int i=1; i<=k;i++)
System.out.print(fibonaci(i)+" ");
}
}

Home Work

Recursive program to find the Sum of the series 1 – 1/2 + 1/3 – 1/4 … 1/N

Given a positive integer N, the task is to find the sum of the series 1 – (1/2) + (1/3) – (1/4) +…. (1/N) using recursion.

Examples:

Input: N = 3
Output: 0.8333333333333333
Explanation:
1 – (1/2) + (1/3) = 0.8333333333333333

Input: N = 4
Output: 0.5833333333333333
Explanation:
1- (1/2) + (1/3) – (1/4) = 0.5833333333333333

i,n,sum

Base condition: if(i>n),sum

else : even: sum=sum-x
odd:sum=sum+x;

Home Work

Recursive Program to print multiplication table of a number

Given a number N, the task is to print its multiplication table using recursion.

Examples

1.Iteration
2.Recursion

Input: N = 5
Output:

```
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

n

Input: N = 8
Output:

```
8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80
```

Recursive function

```
Base cond: if(i>10)
return;
else
SOP("n*i=");
```

Home Work

Recursive program to print formula for GCD of n integers

Given a function gcd(a, b) to find GCD (Greatest Common Divisor) of two number. It is also known that GCD of three elements can be found by gcd(a, gcd(b, c)), similarly for four element it can find the GCD by gcd(a, gcd(b, gcd(c, d))). Given a positive integer n. The task is to print the formula to find the GCD of n integer using given gcd() function.

Examples:

Input : n = 3
Output : gcd(int, gcd(int, int))

Input : n = 5
Output : gcd(int, gcd(int, gcd(int, gcd(int, int))))

2=gcd(b,c)
3=gcd(a,gcd(b,c))
4=gcd(m,gcd(a,gcd(b,c)))

Base Condition:
if(n==1)
return "int"

return gcd(int,r1(n-1))

```java
//

class Recursion5
{
    static String R1(int n)
    {
        if(n==1)
            return "int";

        return "gcd(int,"+R1(n-1)+")";
    }

public static void main(String [] arg)
{
    int n=3;
    System.out.println(R1(n));


}
}
```

```
C  Who can see what you share here? Recording On
Fibonacci:
1 1 2 3 5 8 13 21 34 55 89 144 233 377
7711 28657 46368 75025 121393 196418 31
3524578 5702887 9227465 14930352 241578
80141 267914296 433494437 701408733 113
59680 -811192543 -298632863
C:\ADS>javac Recursion5.java
Recursion5.java:20: error: reached end
}
 ^
1 error

C:\ADS>javac Recursion5.java

C:\ADS>java Recursion5
gcd(int,gcd(int,int))

C:\ADS>
```

R1(2)

gcd(1,R1())

R1(1) → gcd(1,n) ~~gcd(1,)~~

gcd(1,R1(2))

Ackermann's function

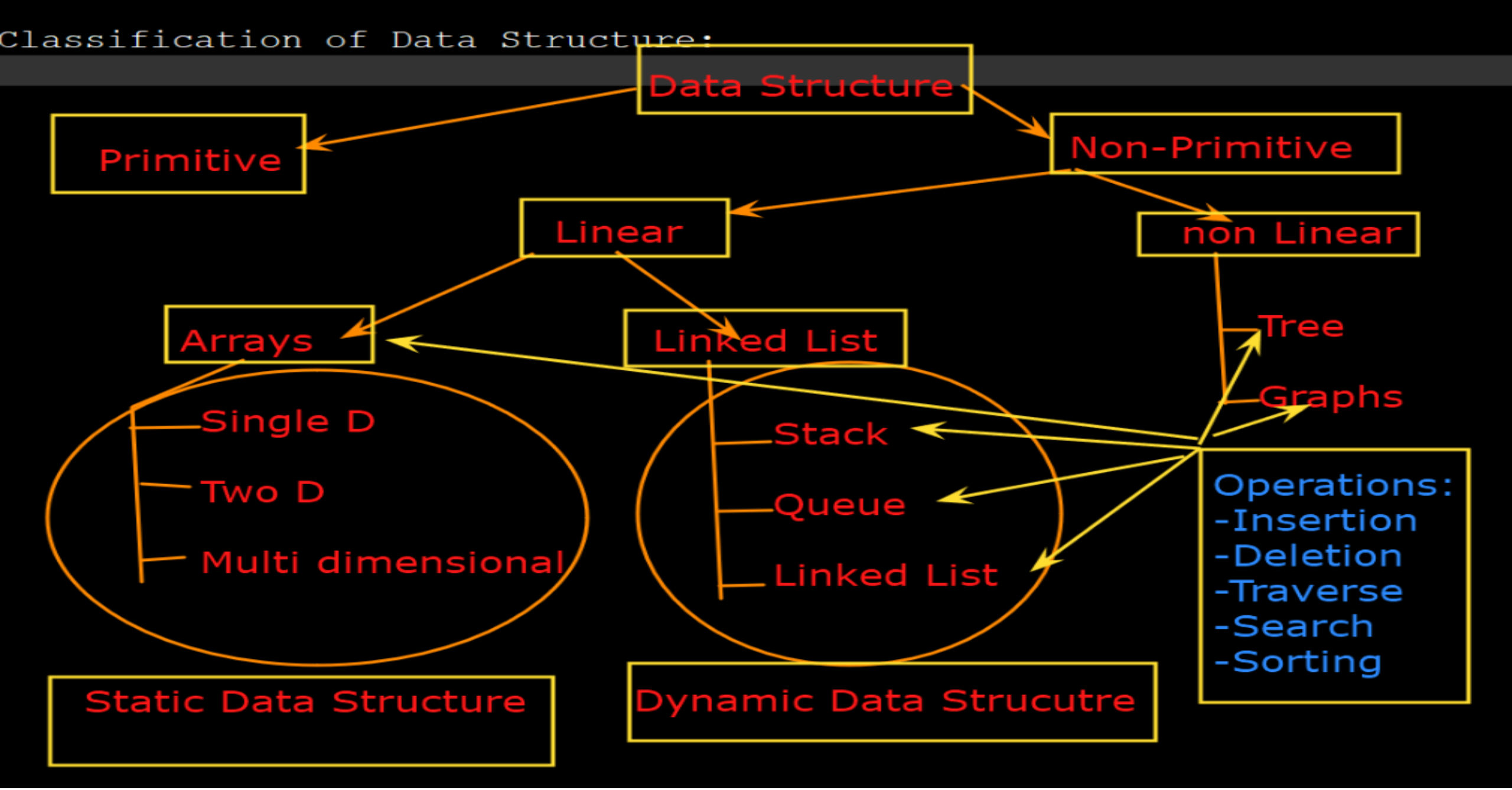
$A(0, n) = n + 1$

$A(m, 1) = A(m+1, 0)$

$A(m+1, n+1) = A(m, A(m+1, n))$

This function build a VERY deep stack very quickly

Arrays


Classification of Data Structure:
Data Structure
Primitive
Non-Primitive
Linear
non Linear
Arrays
Linked List
Tree
Graphs
Single D
Two D
Multi dimensional
Stack
Queue
Linked List
Operations:
-Insertion
-Deletion
-Traverse
-Search
-Sorting
Static Data Structure
Dynamic Data Strucutre

Classification of Data Structure:

Arrays:

Array is finite, ordered and homogeneous collection of elements.

int a[5] =>0.....4

Address (a[i]) = M + (i-L)xw

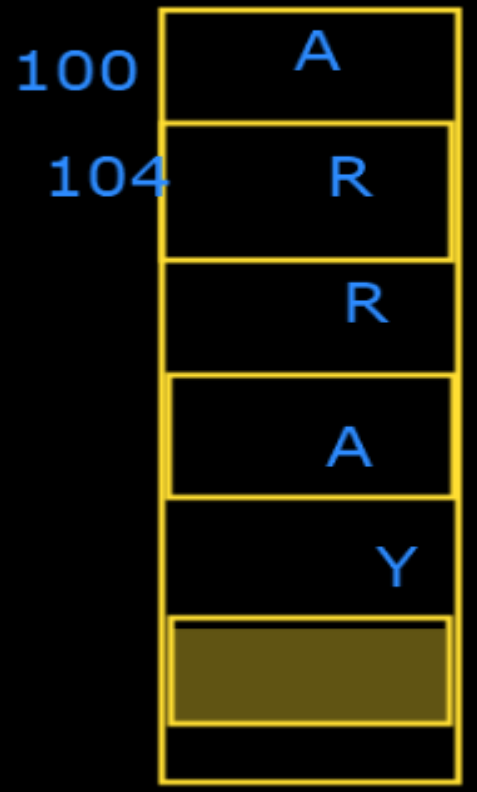
= 100 +(0-0)x4

= 100

Size(a) = U-L+1

= 4-0+1

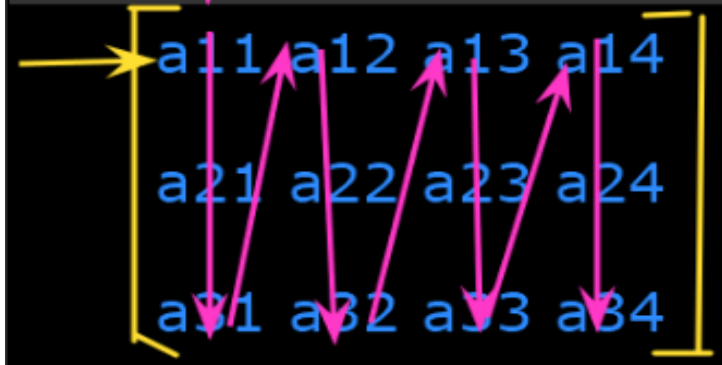
=5



Classification of Data Structure:

Arrays:

Array is finite, ordered and homogeneous collection of elements.



Row Major Order:

address(aij)=M+(i-1)Xn+j-1

a13 =100+(1-1)X4+3-1

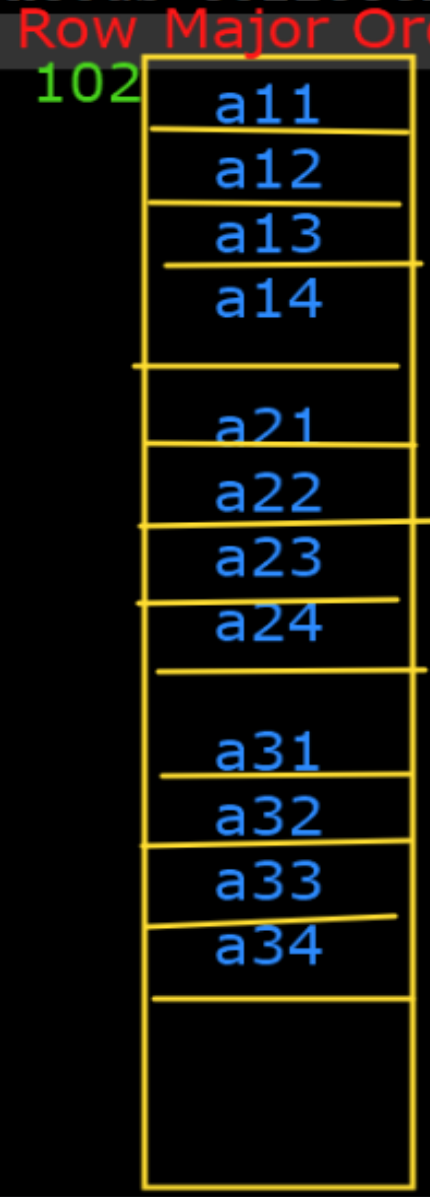
=102

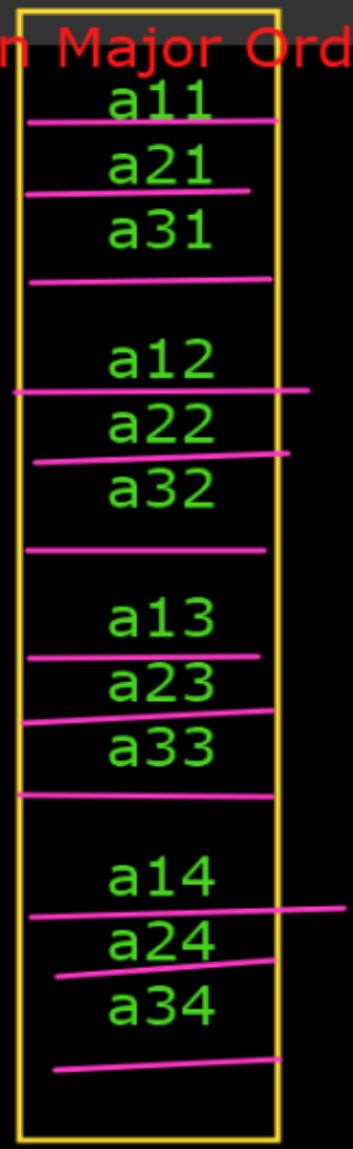
Column Major Order

address(aij)=M+(j-1)xm+i-1

a34 = 100+(4-1)X3+3-1

= 111





```java
        //--Delete
        System.out.println();
        key =55;
        for(j=0;j<n;j++)
        {

            if(a1[j]==key)
                break;

        }
        for(int k=j;k<n;k++)
        {
            a1[k]=a1[k+1];

        }
        n--;

        //--Display
        for(j=0;j<n;j++)
        {
            System.out.print(a1[j]+" ");
        }

    }
}
```

Command Prompt

```
                                ^
1 error

C:\ADS>java Array
22 44 66 77 88 22 55 99 0 11 ..............
Found...

C:\ADS>javac Array.java

C:\ADS>java Array
22 44 66 77 88 22 55 99 0 11 ..............
Found...                      j

22 44 66 77 88 22 99 0 11
C:\ADS>
```