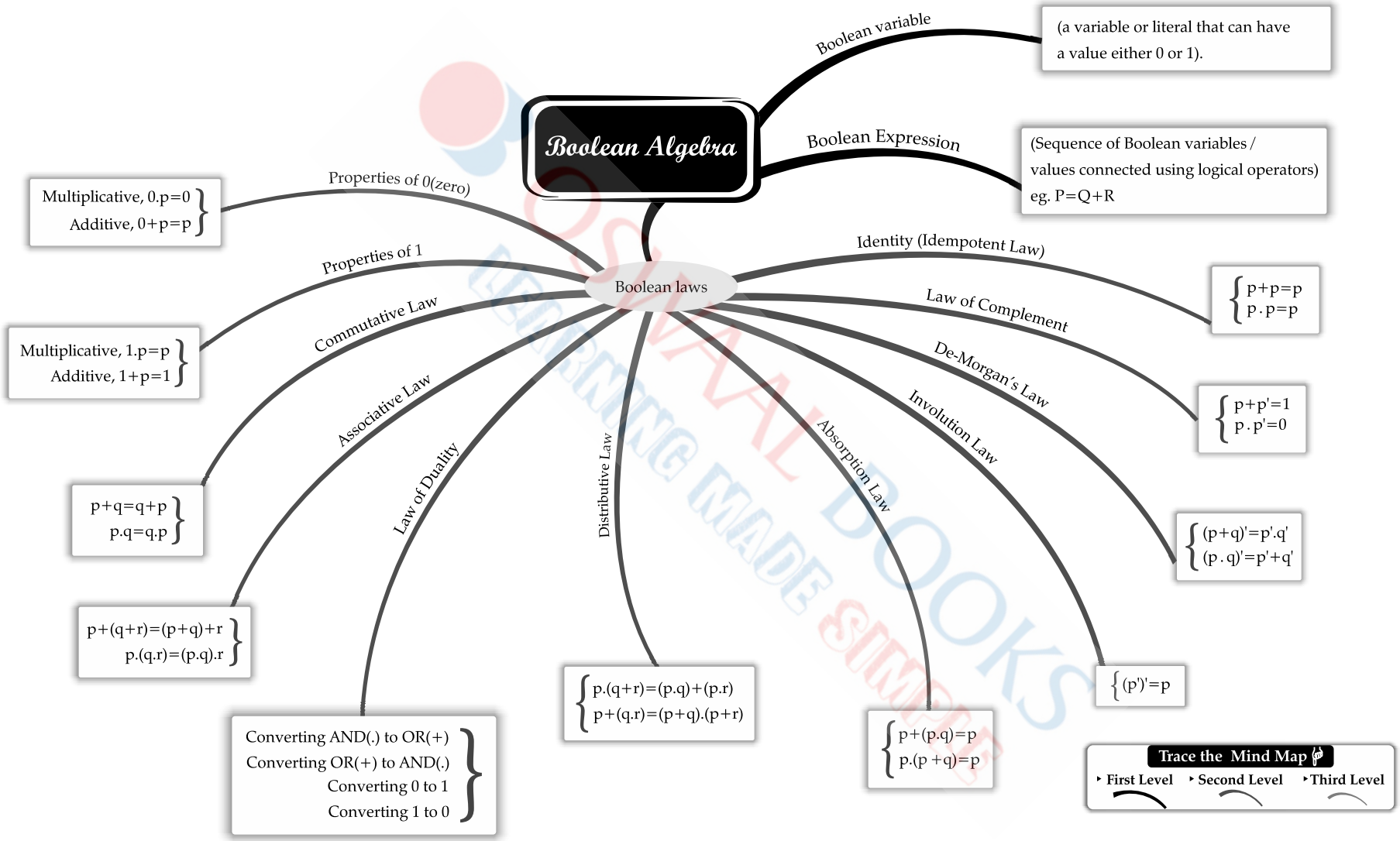
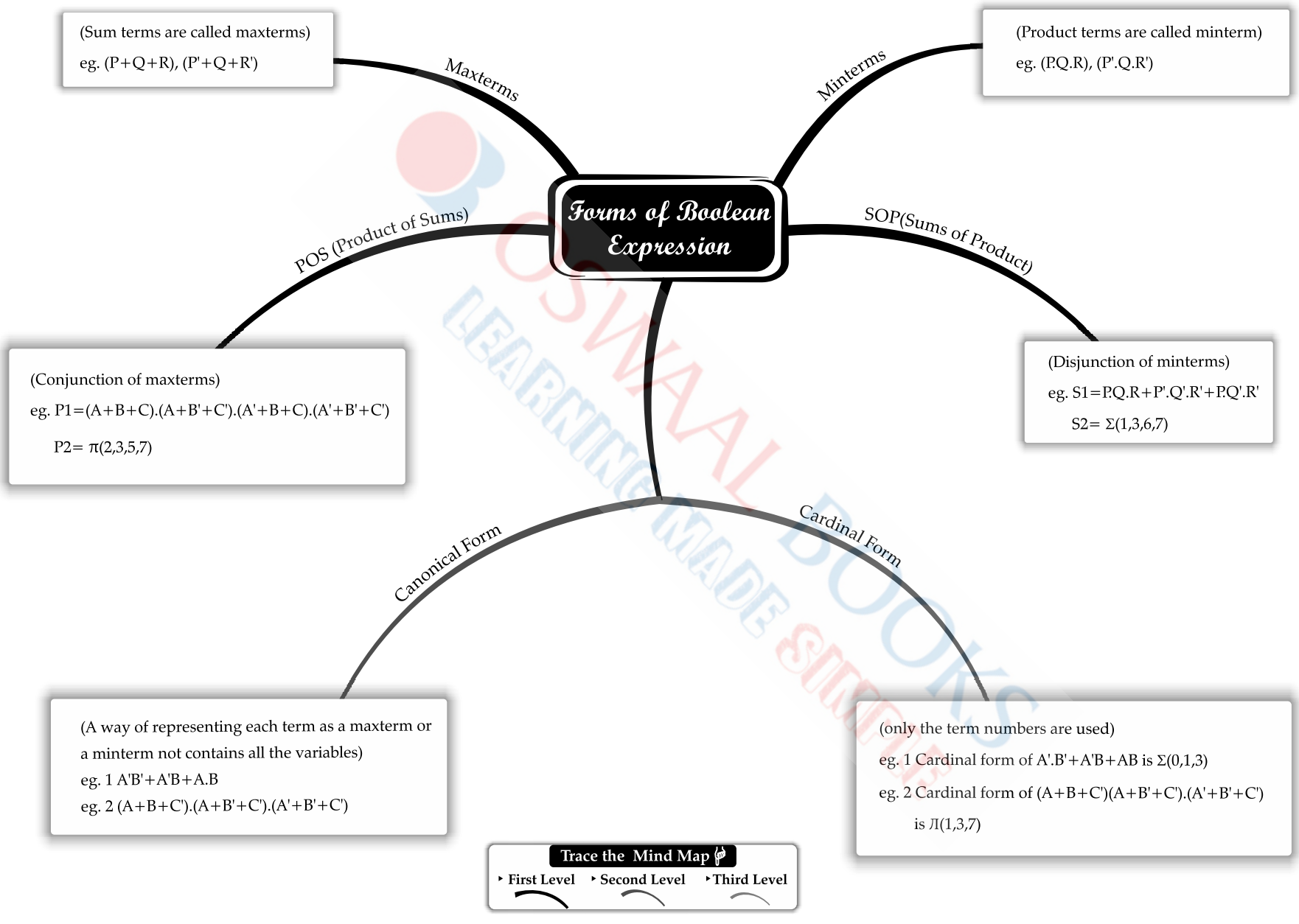
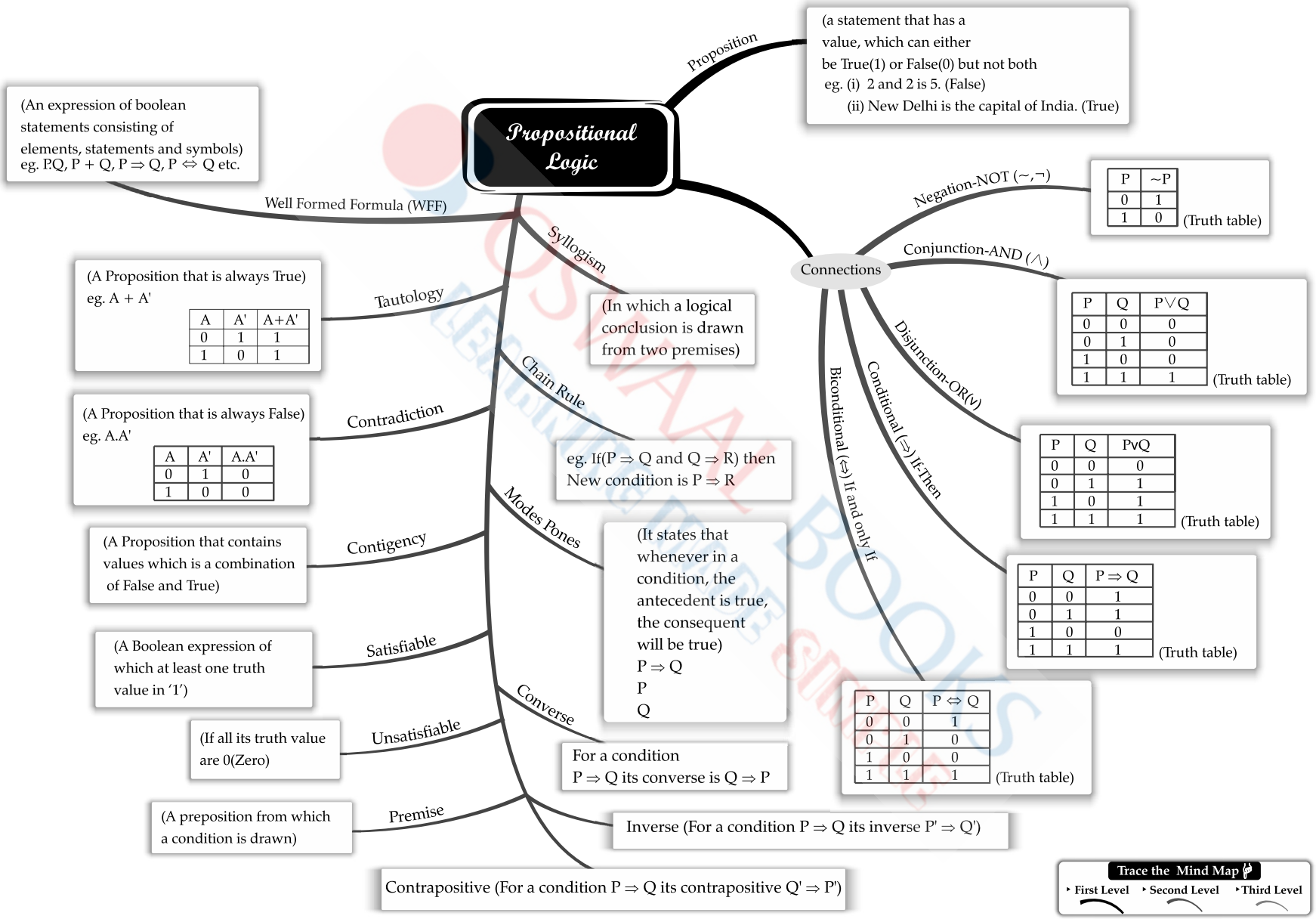
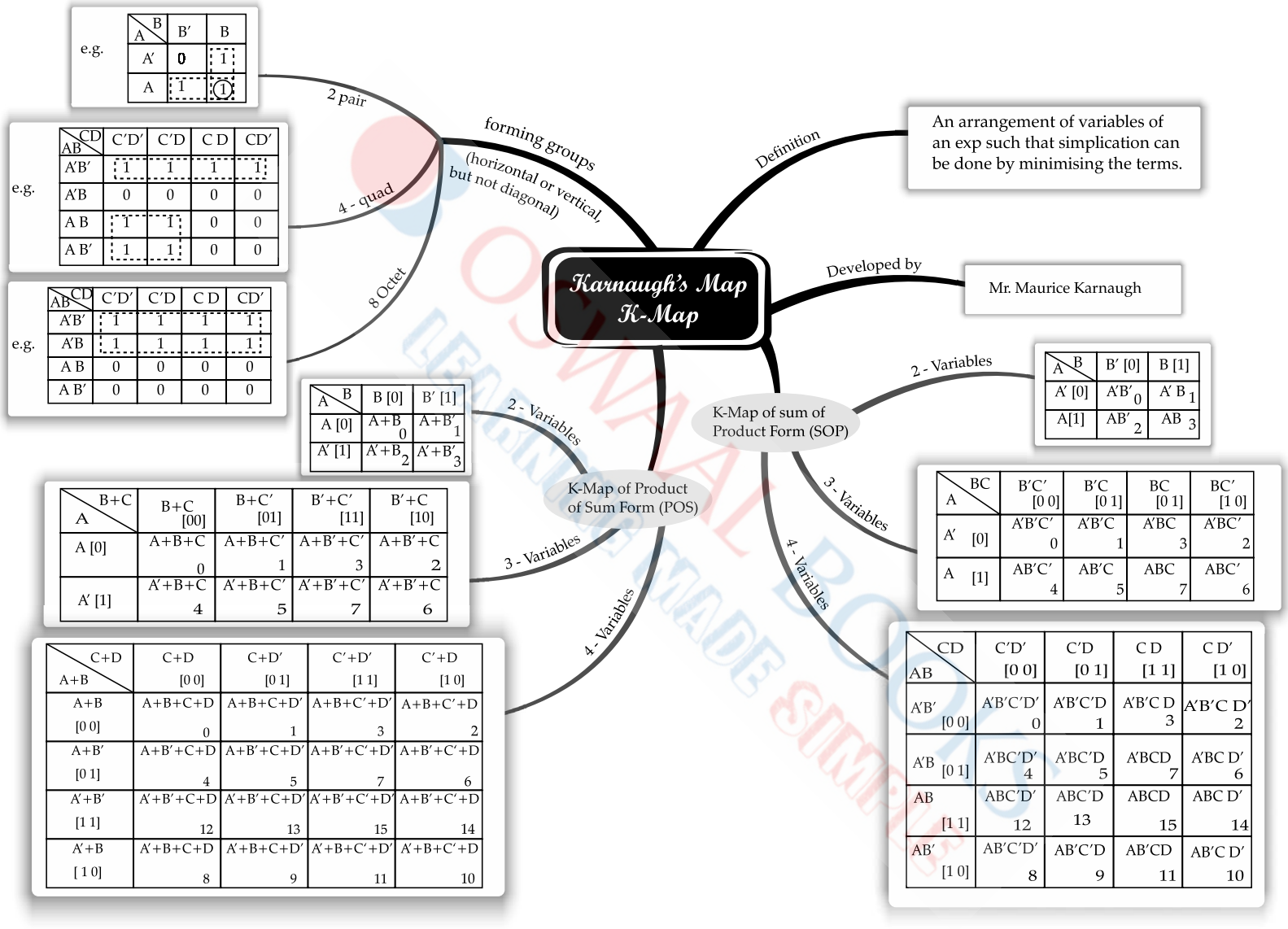


OSWAAL BOOKS LEARNING MADE SIMPLE



OSWAAL BOOKS LEARNING MADE SIMPLE


Propositional Logic
Proposition
(a statement that has a value, which can either be True(1) or False(0) but not both
eg. (i) 2 and 2 is 5. (False)
(ii) New Delhi is the capital of India. (True)
Well Formed Formula (WFF)
(An expression of boolean statements consisting of elements, statements and symbols)
eg. P.Q, P + Q, P ⇒ Q, P ⇔ Q etc.
Connections
Negation-NOT (~,¬)
P ~P
0 1
1 0
(Truth table)
Conjunction-AND (∧)
P Q P∨Q
0 0 0
0 1 0
1 0 0
1 1 1
(Truth table)
Disjunction-OR(∨)
P Q PvQ
0 0 0
0 1 1
1 0 1
1 1 1
(Truth table)
Conditional (⇒) If-Then
P Q P ⇒ Q
0 0 1
0 1 1
1 0 0
1 1 1
(Truth table)
Biconditional (⇔) If and only if
P Q P ⇔ Q
0 0 1
0 1 0
1 0 0
1 1 1
(Truth table)
Syllogism
(In which a logical conclusion is drawn from two premises)
Chain Rule
eg. If(P ⇒ Q and Q ⇒ R) then New condition is P ⇒ R
Modes Pones
(It states that whenever in a condition, the antecedent is true, the consequent will be true)
P ⇒ Q
P
Q
Converse
For a condition P ⇒ Q its converse is Q ⇒ P
Inverse (For a condition P ⇒ Q its inverse P' ⇒ Q')
Contrapositive (For a condition P ⇒ Q its contrapositive Q' ⇒ P')
Tautology
(A Proposition that is always True)
eg. A + A'
A A' A+A'
0 1 1
1 0 1
Contradiction
(A Proposition that is always False)
eg. A.A'
A A' A.A'
0 1 0
1 0 0
Contigency
(A Proposition that contains values which is a combination of False and True)
Satisfiable
(A Boolean expression of which at least one truth value in '1')
Unsatisfiable
(If all its truth value are 0(Zero)
Premise
(A preposition from which a condition is drawn)
Trace the Mind Map
First Level
Second Level
Third Level



A \ B	B′ [0]	B [1]
A′ [0]	A′B′ 0	A′B 1
A [1]	AB′ 2	AB 3

A \ B	B [0]	B′ [1]
A [0]	A+B 0	A+B′ 1
A′ [1]	A′+B 2	A′+B′ 3

A \ BC	B′C′ [0 0]	B′C [0 1]	BC [0 1]	BC′ [1 0]
A′ [0]	A′B′C′ 0	A′B′C 1	A′BC 3	A′BC′ 2
A [1]	AB′C′ 4	AB′C 5	ABC 7	ABC′ 6

A \ B+C	B+C [00]	B+C′ [01]	B′+C′ [11]	B′+C [10]
A [0]	A+B+C 0	A+B+C′ 1	A+B′+C′ 3	A+B′+C 2
A′ [1]	A′+B+C 4	A′+B+C′ 5	A′+B′+C′ 7	A′+B′+C 6

A+B \ C+D	C+D [0 0]	C+D′ [0 1]	C′+D′ [1 1]	C′+D [1 0]
A+B [0 0]	A+B+C+D 0	A+B+C+D′ 1	A+B+C′+D′ 3	A+B+C′+D 2
A+B′ [0 1]	A+B′+C+D 4	A+B′+C+D′ 5	A+B′+C′+D′ 7	A+B′+C′+D 6
A′+B′ [1 1]	A′+B′+C+D 12	A′+B′+C+D′ 13	A′+B′+C′+D′ 15	A+B′+C′+D 14
A′+B [1 0]	A′+B+C+D 8	A′+B+C+D′ 9	A′+B+C′+D′ 11	A′+B+C′+D 10

AB \ CD	C′D′ [0 0]	C′D [0 1]	C D [1 1]	C D′ [1 0]
A′B′ [0 0]	A′B′C′D′ 0	A′B′C′D 1	A′B′C D 3	A′B′C D′ 2
A′B [0 1]	A′BC′D′ 4	A′BC′D 5	A′BCD 7	A′BC D′ 6
AB [1 1]	ABC′D′ 12	ABC′D 13	ABCD 15	ABC D′ 14
AB′ [1 0]	AB′C′D′ 8	AB′C′D 9	AB′CD 11	AB′C D′ 10

e.g.	A \ B	B′	B
	A′	0	1
	A	1	1

e.g.	AB \ CD	C′D′	C′D	C D	CD′
	A′B′	1	1	1	1
	A′B	0	0	0	0
	A B	1	1	0	0
	A B′	1	1	0	0

e.g.	AB \ CD	C′D′	C′D	C D	CD′
	A′B′	1	1	1	1
	A′B	1	1	1	1
	A B	0	0	0	0
	A B′	0	0	0	0

Note : For minimising the terms, First we make all possible octets, them all possible quads and the last all possoble pairs.

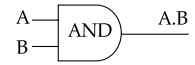
OSWAAL BOOKS
LEARNING MADE SIMPLE

Logic Gates

A small digital circuit used to implement boolean function e.g. AND, OR, NOT etc

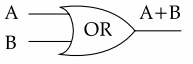
Elementary Logic Gates

AND Gate



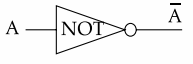
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate



A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT Gate



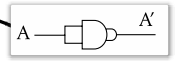
A	$\overline{A}$
0	1
1	0

Universal Gates

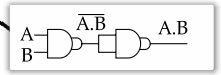
NAND Gate

A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

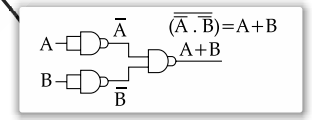
NOT - NAND Gate



AND - NAND Gate



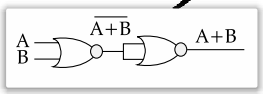
OR - NAND Gate



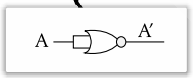
NOR Gate

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

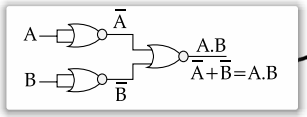
OR Gate by NOR



NOT Gate by NOR

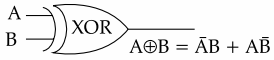


AND Gate by NOR



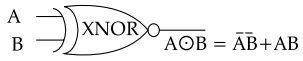
Other specific Logic Gates

Exclusive OR / XOR Gate



A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive NOR / XNOR



A	B	A⊙B
0	0	1
0	1	0
1	0	0
1	1	1

Trace the Mind Map

- First Level
- Second Level
- Third Level

OSWAAL BOOKS
LEARNING MADE SIMPLE

Combinational Circuit

Combining the different gates in a circuit

Examples

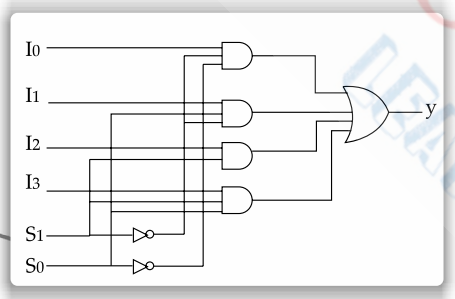
Multiplexer

2n input lines, 1 output line and 'n' select lines

e.g. 4 × 1 Mux.

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Logic diagram

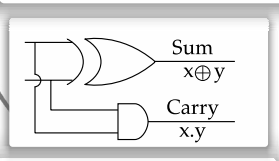


Half Adder

Performs the addition of two bits

x	y	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Logic diagram

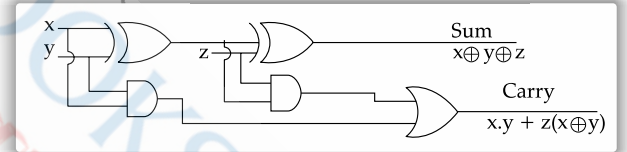


Full Adder

Performs the addition of three bits

x	y	z	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

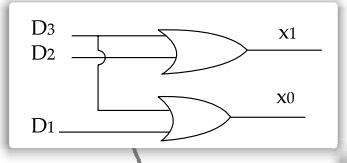
Logic diagram



Encoder

Converts signal from other form to binary form

e.g. 4 × 2 encoder



Logic diagram

D_3	D_2	D_1	D_0	x1	x0
0	0	0	1	0	0
0	0	0	1	0	1
0	1	0	0	1	0
1	0	0	0	1	1

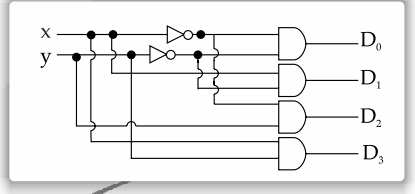
Decoder

Converts signal from Binary form to other form

e.g. 2 × 4 Decoder

x	y	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Logic diagram

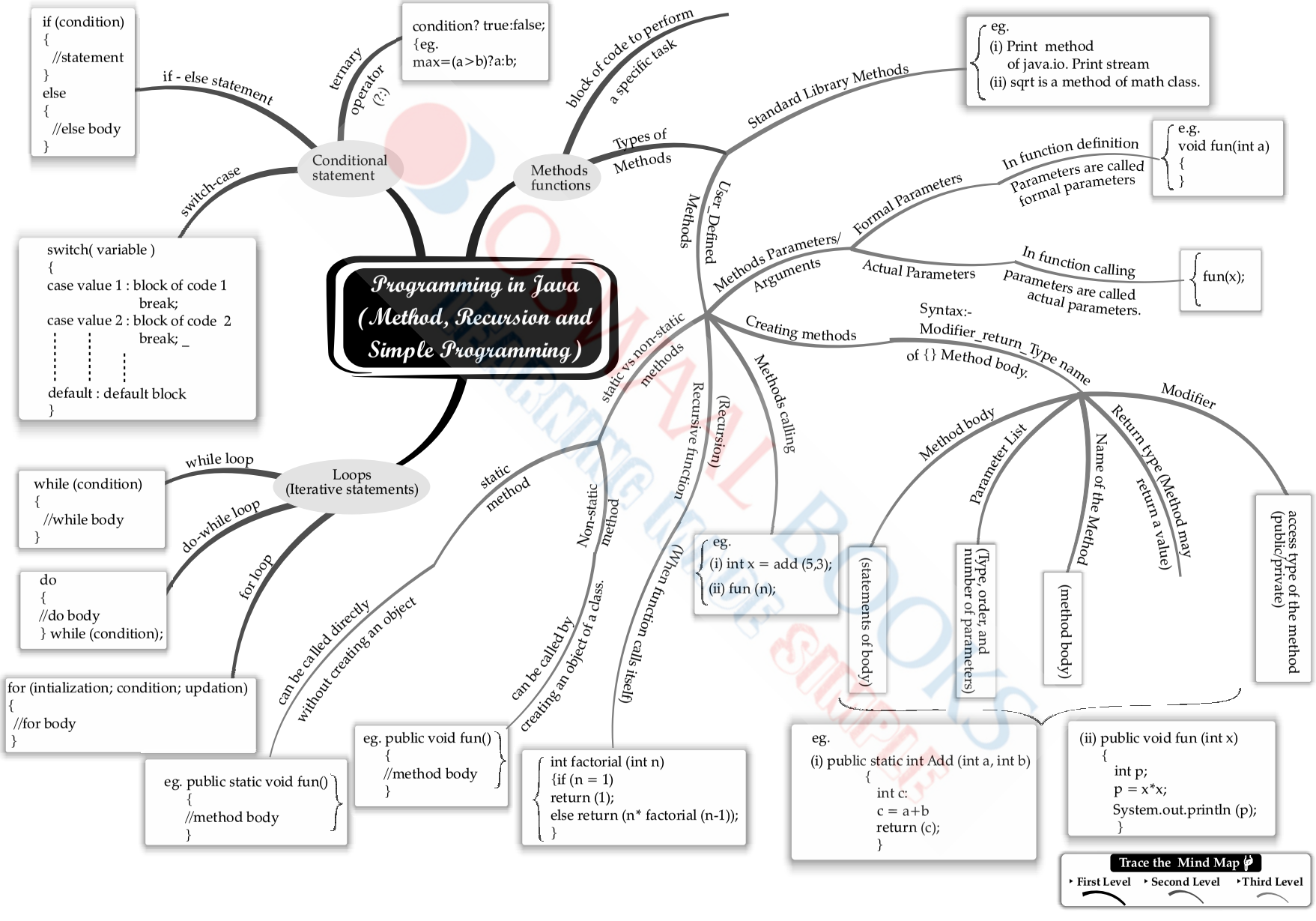


Trace the Mind Map

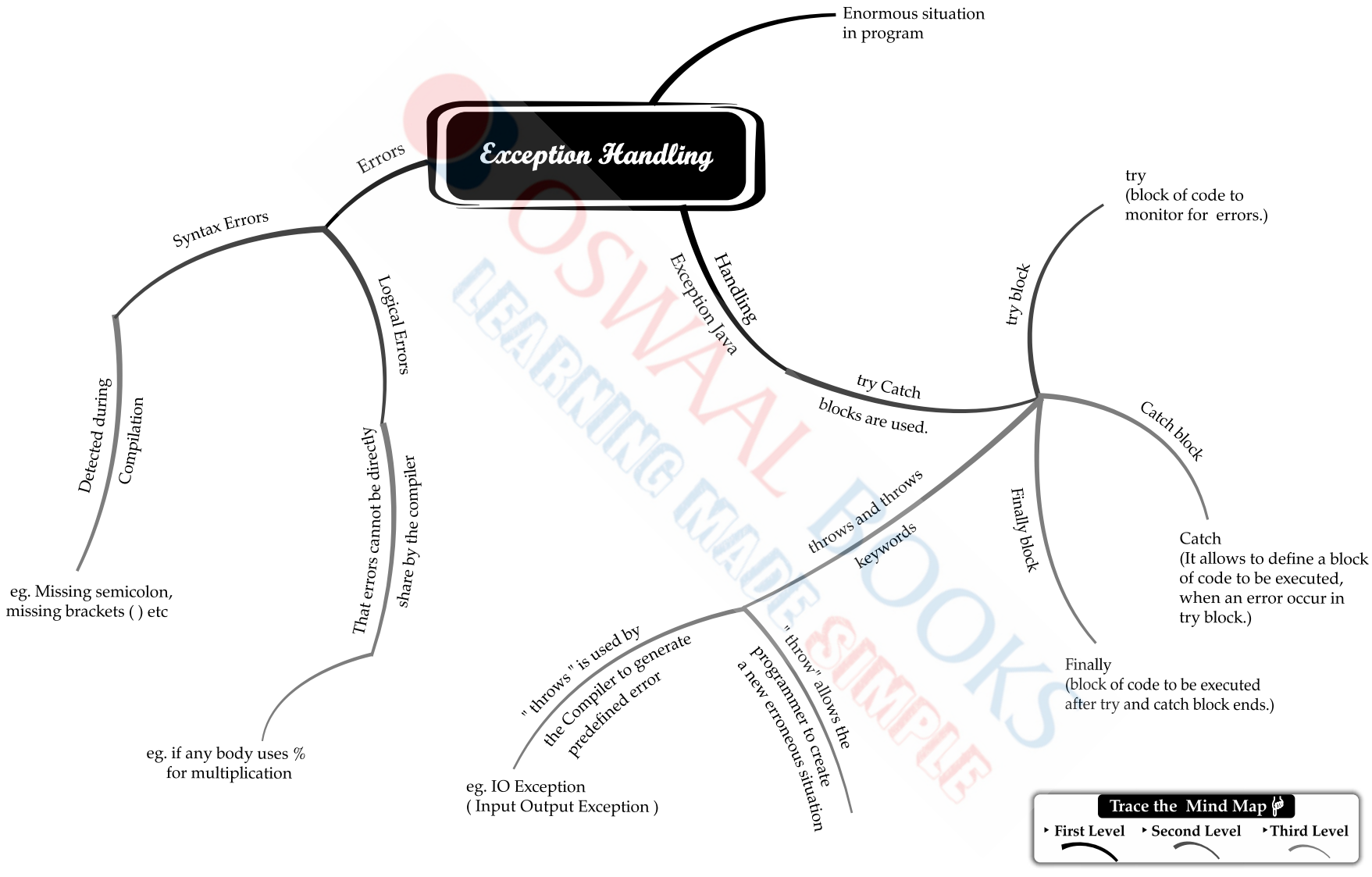
- First Level
- Second Level
- Third Level

OSWAAL BOOKS LEARNING MADE SIMPLE

Programming in Java (Fundamentals)
Features of Java
Platform Independent
due to (Java Byte Code)
Byte Code is independent of the processer of a Computer
"Java Virtual Machine" converts Byte code to Machine Language.
Object Oriented programming
Objects
Class
(instance of a class)
Encapsulation
(blue print that describes the behaviour or state of the object)
Inheritance
(All related data and methods enclosed within one boundary)
Polymorphism
(reusability)
Dynamic Binding
(Same name in many forms)
Simple
Dynamic and Distributed
Robust
Secure
Multithreads
WORA(Write Once Run Any where)
Java Identifiers
Names used for class, variables, and methods
Should begin with a letter (A to Z, a to z) $, underscore(_)
The characters (besides the first) Can be from alpha digit, Underscore, or dollar sign
Keywords cannot be identifiers
eg. a1, $ price, _value, sum.
Keywords
Reserved words
eg. abstract, class, for, try etc.
Variables
Local Variable
declared in methods, constructor, or blocks
eg. class A { fun () { int x; } }
Instance Variables
Class Variables
declared with static keyword in a class but outside a method, constructor, or any block
eg. class A { static int X; }
Data Types
Non- Primitive Data Type
Contains the reference of dynamically created objects
eg. arrays, classes, interfaces.
Primitive Data Type
8 primitive data types defined by the language
eg. byte (1 byte) int (4 bytes)
Type Casting
Assigning a Value of one type to a variable of another type
Wrapper Classes
For each fundamental data type there exist a pre-defined class
to convert simple data types into objects
to convert strings into data type
eg. for 'byte' data type there is 'Byte' wrapper class.
Implicit Type Casting
Java compiler automatically upgrades lower type to higher type.
Explicit Type Casting
to change an entity of one datatype to another by programmer.
eg. double d = 100.5; long l= (long)d;
Trace the Mind Map
First Level
Second Level
Third Level


Programming in Java (Method, Recursion and Simple Programming)
Conditional statement
if - else statement
if (condition)
{
//statement
}
else
{
//else body
}
ternary operator (?:)
condition? true:false;
{eg.
max=(a>b)?a:b;
switch-case
switch(variable)
{
case value 1 : block of code 1
break;
case value 2 : block of code 2
break; _
default : default block
}
Loops (Iterative statements)
while loop
while (condition)
{
//while body
}
do-while loop
do
{
//do body
} while (condition);
for loop
for (intialization; condition; updation)
{
//for body
}
Methods functions
block of code to perform a specific task
Types of Methods
Standard Library Methods
eg.
(i) Print method
of java.io. Print stream
(ii) sqrt is a method of math class.
User Defined Methods
Methods Parameters/ Arguments
Formal Parameters
In function definition
Parameters are called formal parameters
e.g.
void fun(int a)
{
}
Actual Parameters
In function calling
parameters are called actual parameters.
fun(x);
Creating methods
Syntax:-
Modifier_return_Type name of {} Method body.
Modifier
access type of the method (public/private)
Return type (Method may return a value)
Name of the Method
(method body)
Parameter List
(Type, order, and number of parameters)
Method body
(statements of body)
eg.
(i) public static int Add (int a, int b)
{
int c:
c = a+b
return (c);
}
(ii) public void fun (int x)
{
int p;
p = x*x;
System.out.println (p);
}
Methods calling
eg.
(i) int x = add (5,3);
(ii) fun (n);
Recursive function (Recursion)
(When function call itself)
int factorial (int n)
{if (n = 1)
return (1);
else return (n* factorial (n-1));
}
static vs non-static methods
static method
can be called directly without creating an object
eg. public static void fun()
{
//method body
}
Non-static method
can be called by creating an object of a class.
eg. public void fun()
{
//method body
}
Trace the Mind Map
First Level
Second Level
Third Level

Programming in Java (Arrays)
One dimensional array
Array declaration
Data_type array_name [ ] [ ] = new Data_type [row_size] [col_size];
eg. int TA [ ] [ ] = new int [4] [5];
Storage of elements
Row-major
A[p][q] = B + W {(p –Lr) } + (q–Lc)};
Column major
A[p][q] = B + W {tc (p–Lr)} + {(q–Lc)};
A[p] [q] @Address at row p and col q.
B @Base address
W@storage space in bytes
Lr@Lower index of the Row
Ur@Upper index of the Row
Lc@Upper index of the col.
Uc@Upper index of the col.
tc@Total no. of col. (Uc-Lc+1)
tr@Total no. of rows (Ur-Lr+1)
Two dimensional array
Array
Collection of similar types of elements
Array declaration
Data_type array_name [ ] = new Data_type [size];
eg.
(i) int A [ ] = new int [10];
(ii) double B [ ] = {1.2, 3.5, 6.2, 1.8};
Storage of elements in an array
eg. int A [ ] = {5,12,8,17,10};
A
5 12 8 17 10
0 1 2 3 4
Array Assignment
eg. (i) int A [ ] = new int [5];
A [0] = 6; A [1] = 8;
int B [ ] = new int [5];
(ii) scanner sc = new scanner (System.in);
System.out.println ("Enter 5 numbers");
for (int i = 0; i<5; i++)
{A [i] = sc.nextInt ();}
Array Operations
Searching an element in an array
Linear search
Binary Search
Sorting an array
Bubble sort
Insertion sort
Selection sort
Finding out maximum and minimum element in an array
Trace the Mind Map
First Level
Second Level
Third Level



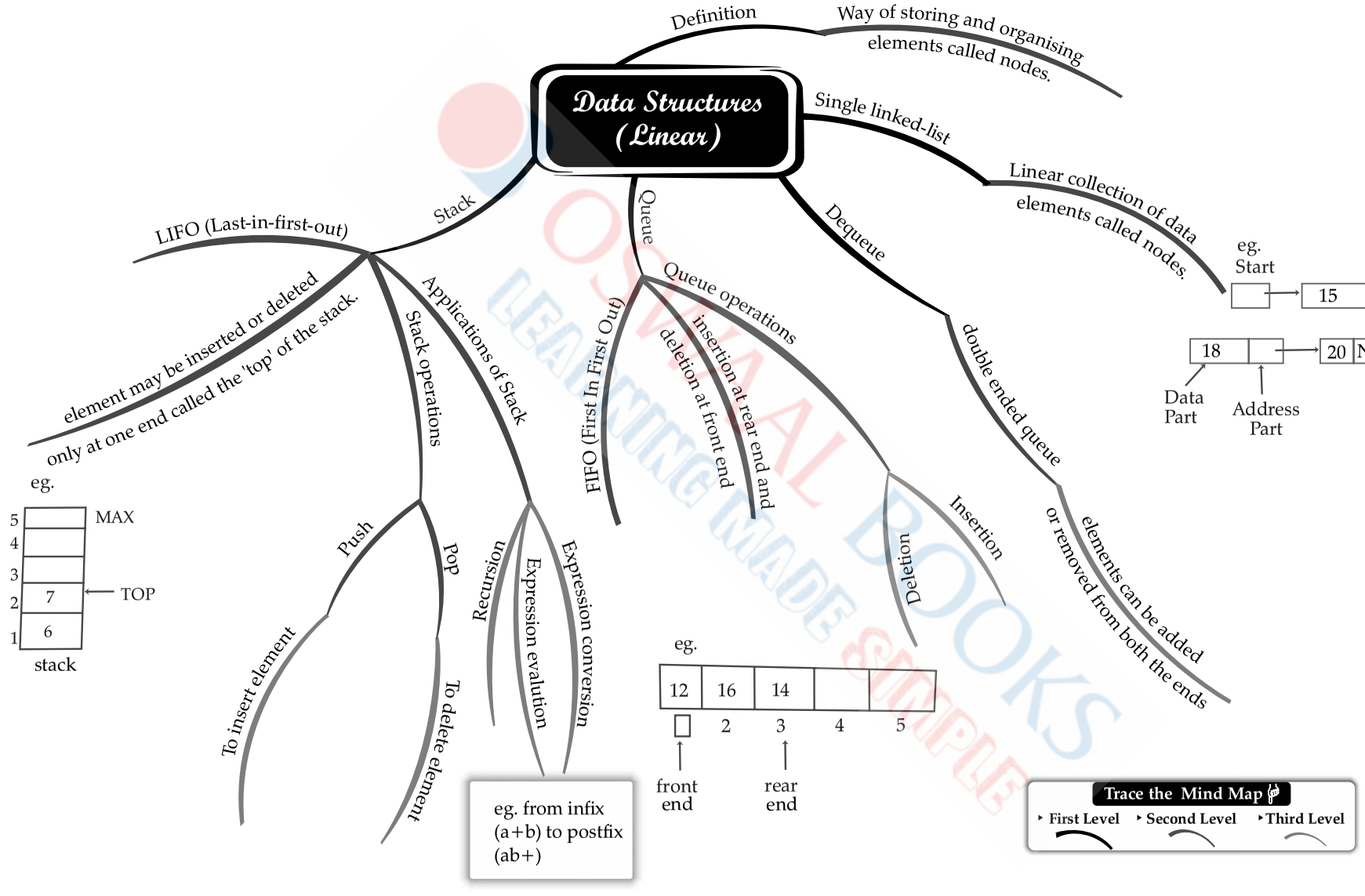
OSWAAL BOOKS LEARNING MADE SIMPLE

Object Oriented Pragaming (Polymorphism Data Inheritance)

- Object
  - Instant of a class
    - eg. Adder x= new Adder ( );
  - blue print that describes the behavior or the state of the Object.
- Class
- Polymorphism
  - Same name in different forms
    - Over loading Constructor (Compile time)
      - e.g. class A
        A ( ) { }
        A (int o)
        { }
    - Function Over loading (Compile time)
      - e.g. (i) void fun (int a) { }
        (ii) void fun (int a, int b) { }
  - Multiple inheritance Give away to apply
    - eg. Public interface XY
      public abstract
      and fun ( ); { }
- Interface
  - Contain only Object method and static final date
  - Abstract class
    - Can be inherited
      - Can not have Objects.
      - Objects of subclass can use Base - abstract methods.
  - Constructors
    - Subclass cannot inherit the constructor of a Base class
    - Default constructor of a base class is executed automatically at the time of creating the objects of a sub -class .
    - The parameters of constructor Super class are assigned from the Sub- class.
  - Overriding (Run time poly morphism
    - Declare a method in Subclass which is already Present in the base - class.
      - For base class method, Super keyword is used.
      - Object of sub class can only instantiate its methods.
  - Types of Inheritance
    - Allows the class to share properties of class.
      - Class A { } Base class
      - Class B extends A { } Sub class -ob A.
    - Single Inheritance
      - X → Base class
      - Y → Sub / Derived class
    - Multi level Inheritance
      - X → Base class of Y
      - Y → Sub class of X, Base class of Z
      - Z → Sub class of Y
    - Multiple Inheritance
      - Base classes: X, Y → Z
      - Not supported by java
    - Hierarchical Inheritance
      - W → Base class
      - X, Y → subclass
      - Z

Trace the Mind Map
- First Level
- Second Level
- Third Level

OSWAAL BOOKS LEARNING WITH SIMPLICITY


Data Structures (Linear)
Definition
Way of storing and organising elements called nodes.
Single linked-list
Linear collection of data elements called nodes.
eg.
Start
15
18
20
NULL
Data Part
Address Part
Stack
LIFO (Last-in-first-out)
element may be inserted or deleted only at one end called the 'top' of the stack.
eg.
5
4
3
2
1
7
6
MAX
TOP
stack
Stack operations
Push
Pop
To insert element
To delete element
Applications of Stack
Recursion
Expression evalution
Expression conversion
eg. from infix (a+b) to postfix (ab+)
Queue
FIFO (First In First Out)
Queue operations
insertion at rear end and deletion at front end
eg.
12
16
14
2
3
4
5
front end
rear end
Dequeue
double ended queue
elements can be added or removed from both the ends
Insertion
Deletion
Trace the Mind Map
First Level
Second Level
Third Level