CHAPTER-1

NUMBERS

TOPIC-1
Number System

Revision Notes

- Numbers are used for counting and data representation. They are made from symbols, which are called digits. Digits are finite but numbers are infinite.

➢ **Number System**A set of symbolic values used to represent different quantities in numerical form is known as number system.

- When we enter data into the computer, it translates them into binary numbers as computer can understand only 0 and 1.
- Base of a number system states how many symbols are present for use in this number system.
- The most common positional number systems used are decimal, binary, octal and hexadecimal systems, in which decimal number system is used in general our day to day life.

The number systems in different bases are described below:

(a) **Binary Number System :** Computers use binary number system for counting and arithmetic operation. This system contains only two digits i.e. 0 and 1 so it has base 2.

e.g. $(1101)_2$, $(011)_2$, $(1001)_2$, etc.

(b) **Octal Number System :** This number system has a base of 8, as it uses 8 symbolic values 0, 1, 2, 3, 4, 5, 6, and 7. Thus each digit of an octal number can have any value from 0 to 7.

e.g. $(237)_8$, $(1534)_8$ etc.

(c) **Decimal Number System :** The number system that we use in our daily life is the decimal number system. It has base 10 as it uses 10 digits from 0 to 9 (i.e. 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9).

e.g. $(239)_{10}$, $(756)_{10}$ etc.

(d) **Hexadecimal Number System :** In this number system, the base or radix is 16. Thus, it has 16 symbols. It uses the digits 0 to 9 (0, 1, 2, 3, 4, 5, 6, 7, 8 and 9) and letters A, B, C, D, E and F (with respected value to 10, 11, 12, 13, 14 and 15).

e.g. $(12AB)_{16}$, $(43DE)_{16}$ etc.

- The numbers are generated by reusing the digits in combination, when single digits are exhausted.
- In various number systems, 10 appears at different levels depending on the number of digits present in that number system.
- In binary number system, digits exhaust after 1[01]. Thus, 10 appears after 1. So, the sequence of numbers are:

  0, 1, 10, 11, 100, 101, 110, 111, 1000, ……. etc.
- In octal number system, digits exhaust after 7 [0 1 …….7]. Thus 10 appears after 7. So, the sequence of numbers is:

  1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, …. etc.
- In decimal number system, digits exhaust after 9 [01…..9].

  Thus 10 appears after 9. So the sequence of numbers is :

  1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, …. etc.
- In hexadecimal number system, digits exhaust after F [0 1 ….. 8 9 A B … E F]. Thus, 10 appear after F. So, the sequence of numbers is : 0,1,2,3 . . . 8,9 A,B . . . F,10,11,12,13, …. etc.

➢ **Number Chart**

Binary	Octal	Decimal
0	0	0
1	1	1

10	2	2
11	3	3
100	4	4
101	5	5
110	6	6
111	7	7
1000	10	8
1001	11	9
1010	12	10
1011	13	11
1100	14	12
1101	15	13
1110	16	14
1111	17	15
10000	20	16
10001	21	17
10010	22	18
10011	23	19
10100	24	20

➢ **Number System Conversion**

- Conversion from one number system to another number system is required because the user enters the data into the computer in decimal number system only.
- These conversions are described below:

(i) Decimal to Binary

- To convert an integer decimal number to binary, divide the decimal number continuously by 2. The corresponding binary number can be obtained by tracing the remainders in reverse order *i.e.* from MSB + LSB.

e.g. $(43)_{10} \rightarrow (?)_2$

		Remainder
2	43	1→LSB *i.e* least significant bit
2	21	1
2	10	0
2	5	1
2	2	0
2	1	1→MSB *i.e* most significant bit
	0	

Then $(43)_{10} \rightarrow (101011)_2$

- Conversion of fractional part is done by continuous multiplication of the fractional part by 2. In this process, the integer generated (either 0 or 1) is taken aside and the rest of the fractional part is multiplied by 2. The process stops when the fractional part's result is 0 or becomes recursive.

e.g. $(0.25)_{10} \rightarrow (?)_2$

	Integer part
$0.25 \times 2 = 0.50$	0
$0.50 \times 2 = 1.00$	1

$(0.25)_{10} \rightarrow (0.01)_2$

(ii) Decimal to Octal

- To convert an integer decimal number to octal, divide the decimal number continuously by 8 till it becomes 0. The corresponding octal number can be obtained by tracing the remainders in reverse order.
- To convert the fractional part, multiply it with 8. Keep aside the integer generated (between 0 to 7) and next fractional part is multiplied by 8. The process stops when the fractional part's result is 0 or the value becomes recursive.

e.g. $(90.25)_{10} \rightarrow (?)_8$

For Integer part,

		Remainder	
8	90	2	↑ LSB
8	11	3	
8	1	1	
	0		MSB

For fractional part,	Integer part
$0.25 \times 8 = 2.00$	2
$0.00 \times 8 = 0$	

So, $(90.25)_{10} = (132.2)_8$

(iii) Decimal to Hexadecimal

- To convert an integer decimal number to hexadecimal, divide the decimal number continuously by 16 till it becomes 0. The corresponding hexadecimal number can be obtained by tracing the remainders in reverse order.
- To convert the fractional part, multiply it with 16. Keep aside the integer generated (between 0 to F) and process stops when the fractional part result is 0 or the value becomes recursive.

e.g. $(100.25)_{10} \rightarrow (?)_{16}$

For integer part,

		Remainder	
16	100	4	↑
16	6	6	
	0		

For fractional part,

$0.25 \times 16 = 4.00$

$0.00 \times 16 = 0$

So, $(100.25)_{10} = (64.4)_{16}$

(iv) Binary to Decimal

- To convert an integer binary number to decimal, multiply the digits of the binary number of 2^n, where n begins with 0 for the digit in unit's place and increase by 1 each time for the next digit with higher place value. Finally, add all these products.
- To convert the fractional part of a binary number to decimal, multiply the digits that appear after the decimal point by 2^p, where p begins with –1 and goes on decreasing by 1.

e.g. $(1101.10)_2 \rightarrow (?)_{10}$

$$\begin{aligned}(1101.10)_2 &= (1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0) + (1 \times 2^{-1} + 0 \times 2^{-2})\\ &= (1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1) + (1 \times 0.5 + 0 \times 0.25)\\ &= 13 + 0.5\end{aligned}$$

$(1101.10)_2 = (13.5)_{10}$

(v) Octal to Decimal

- To convert an integer octal number to decimal, multiply the digits of the octal number by 8^n, where n begins with 0 for the digit in unit's place and increases by 1 each time for the next digit with higher place value. Finally, add all the products.
- The process of converting the fractional part of an octal number to decimal is by multiplying the digits that appear after the decimal point by 8^p, where p begins with–1 and goes on decreasing by 1.

e.g. $(402.12)_8 = (?)_{10}$

$$\begin{aligned}(402.12)_8 &= (4 \times 8^2 + 0 \times 8^1 + 2 \times 8^0) + (1 \times 8^{-1} + 2 \times 8^{-2})\\ &= (4\times64 + 0 \times 8 + 2 \times 1) + (1 \times 0.125 + 2 \times 0.015625)\\ &= 256 + 0 + 2 + 0.125 + 0.03125\end{aligned}$$

$(402.12)_8 = (258.15625)_{10}$

(vi) Hexadecimal to Decimal

- To convert an integer hexadecimal number to decimal, multiply the digits of the hexadecimal number by 16^n, where n begins with 0 for the digit in unit's place and increases by 1 each time for the next digit with higher place value. Finally, add all these products.
- The process of converting the fractional part of a hexadecimal number to decimal is by multiplying the digits that appear after the decimal point by 16^p, where p begins with –1 and goes on decreasing by 1.

e.g. $(100B\,.\,A)_{16} = (?)_{10}$

$$(100B\,.\,A)_{16} = (1 \times 16^3 + 0 \times 16^2 + 0 \times 16^1 + B \times 16^0) + (A + 16^{-1})$$
$$= (1 \times 4096 + 0 \times 256 + 0 \times 16 + 11 \times 1) + (10 \times 0.0625)$$
$$= 4096 + 0 + 0 + 11 + 0.625$$
$$(100B\,.\,A)_{16} = (4107.625)_{10}$$

(vii) Binary to octal

- Numbers systems, octal ($8 = 2^3$) and binary ($2 = 2^1$) are related in such a way that each octal digit can be replaced by corresponding three binary digits in sequential group.
- To convert a binary number to octal, three digits of the binary number have to be grouped and replaced by one octal digit. The digit grouping takes place from right side. Leading zeros can be added if required.
- To convert the fractional part, three digits of the binary number have to be grouped and replaced by one octal digit where grouping starts from the point's position.

e.g. $(1110100.1011)_2 \rightarrow (?)_8$

$$\begin{array}{ccccc} \underline{001} & \underline{110} & \underline{100} & .\underline{101} & \underline{100} \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 6 & 4 & 5 & 4 \end{array}$$

So, $(1110100.1011)_2 \rightarrow (164.54)_8$

(viii) Binary to Hexadecimal

- Number systems, hexadecimal ($16 = 2^4$) and binary ($2 = 2^1$) are related in such a way that each hexadecimal digit can be replaced by corresponding four binary digits sequential group.
- To convert a binary number to hexadecimal, four digits of the binary number have to be grouped and replaced by one hexadecimal digit. The digit grouping take place from the right side. Leading 0's can be put if required.
- To convert the fractional part, four digits of the binary number have to be grouped and replaced by one hexadecimal digit, where grouping starts from the decimal point position. Subsequent zeros can be put if required.

e.g. $(1111001010.1101101)_2 = (?)_{16}$

$$\begin{array}{cccccc} \underline{0011} & \underline{1100} & \underline{1010} & . & \underline{1101} & \underline{1010} \\ \downarrow & \downarrow & \downarrow & & \downarrow & \downarrow \\ 3 & 12 = C & 10 = A & & 13 = D & 10 = A \end{array}$$

So, $(1111001010.1101101)_2 = (3CA\,.\,DA)_{16}$

(ix) Octal to Binary

- One octal digit can replaced corresponding three binary digits using the number chart.
- An octal number can be converted to binary by replacing each digit to the corresponding three digit binary number.

e.g. $(143.52)_8 = (?)_2$

$$\begin{array}{lcccccc} (143.52)_8 \rightarrow & 1 & 4 & 3 & . & 5 & 2 \\ = & \underline{001} & \underline{100} & \underline{011} & . & \underline{101} & \underline{010} \end{array}$$

$(143.52)_8 = (001100011.101010)_2$

(x) Hexadecimal to Binary

- One hexadecimal digit can replace corresponding four binary digits using the number chart.
- Hexadecimal number can be converted to binary by replacing each digit to the corresponding four digit binary number.

e.g. $(423.A1)_{16} = (?)_2$

$$\begin{array}{lcccccc} (423.A1)_{16} = & 4 & 2 & 3 & . & A(10) & 1 \\ = & \underline{0100} & \underline{0010} & \underline{0011} & & \underline{1010} & \underline{0001} \end{array}$$

$(423.A1)_{16} = (010000100011.10100001)_2$

(xi) Octal to Hexadecimal

- Conversion from octal to hexadecimal can be done in two ways:
- Convert the octal number to binary then convert the binary number to hexadecimal.
- Convert the octal number to decimal then convert the decimal number to hexadecimal.

e.g. $(42.6)_8 = (?)_{16}$

$$\begin{array}{lcccc} (42.6)_8 = & 4 & 2 & . & 6 \\ = & 100 & 010 & . & 110 \end{array}$$

$= (100010.110)_2 = \underline{0010}\ \underline{0010}\ .\ \underline{1100}$

$= (22.C)_{16}$

(xii) Hexadecimal to Octal

Conversion from hexadecimal to octal can be done in two ways

- Convert the hexadecimal number to decimal then convert decimal number to octal.
- Convert the hexadecimal number to binary then convert the binary number to octal.

e.g. $(4D1.9)_{16} = (?)_8$

$(4D1.9)_{16} =$ 4 D (13) 1 . 9

$= \underline{0100}\ \ \underline{1101}\ \ \underline{0001}\ \ \underline{1001}$

$= (010011010001.1001)_2$

$(010011010001.1001)_2 = \underline{010}\ \underline{011}\ \underline{010}\ \underline{001}\ .\ \underline{100}\ \underline{100}$

$(4D1.9)_{16} = (2321.1.44)_8$

Know the Terms

- **Decimal point :** The point which is used to separate the integer and fractional part of the number, is called decimal point.
- **Most Significant Bit(MSB) :** The leftmost bit carries the largest weight and hence, is called the most significant bit.
- **Least Significant Bit (LSB) :** The rightmost bit carries the smallest weight and hence, is called the least significant bit.
  - Bit or binary digit are the symbol 0 or 1.
  - MSD-Most Significant Digit.
  - **LSD-Least Significant Digit.**

TOPIC-2
Arithmetic Operations

Revision Notes

Binary Arithmetic

- In binary number system, there are only 2 digits 0 and 1, and any number can be represented by these two digits.
- The arithmetic of binary number means the operation of addition, subtraction, multiplication, and division. Binary arithmetic is essential part of all the digital computers and many other digital system.

(i) Binary Addition

- In any number system, the concept of addition states that adding unit value (1) to any number returns the next number in the number line.

There are four rules of binary addition

S.X.	A+B	Sum	Carry
(i)	0+0	0	0
(ii)	0+1	1	0
(iii)	1+0	1	0
(iv)	1+1	0	1

In fourth case, a binary addition is creating a sum of (1+1=10) i.e. 0 is written in given column and carry 1 to the next column.

e.g.

```
      1  1        ← carry
      1  1  1  0  1
+           1  0  0
-------------------
   1  0  0  0  0  1
```

- **Addition of Fractions**
  - Make both the numbers have equal number of digits by putting leading and following zeros.

- Start addition from the fractional part without disturbing the position of the point.

```
e.g.   111001 . 1101 + 1101 . 10
carry  →     11  11
             111001 . 1101
           + 001101 . 1000
           ----------------
            1000111 . 0101
           ----------------
```

(ii) Binary subtraction

There are three methods of performing binary subtraction:

(a) Method of Borrow : In any number system, the concept of subtraction states that subtracting unit value (1) from any number returns the previous number in the number line.

There are four rules of binary subtraction:

A – B	Subtract	Borrow
0 – 0	0	0
1 – 0	1	0
1 – 1	0	0
0 – 1	1	1

e.g. $(1\,0\,1\,0)_2 - (1\,1\,1)_2$

```
   1 0 1 0
 –   1 1 1
 ---------
   0 0 1 1
 ---------
```

(b) Method of One's Complement : One's complement of a binary number is found by inverting the zeros and ones, without changing the number of digits.

The steps to be followed in subtraction by 1's complement are:

- To write down 1's complement of the subtrahend
- To add this with the minuend
- If the result of addition has a carry over then it is dropped and an 1 is added in the last bit.
- If there is no carry over, then 1's complement of the result of addition is obtained to get the final result and it is negative.

```
e.g.      101011 – 111001
          1's complement of 111001 is 000110
              Minuend        101011
          1't complement    000110
                            ------
                            001101
                            ------
```

Here there is no carry, answer is - (1's complement of the sum obtained 110001)
i.e. **001110**

(c) Method of Two's Complement

Two's complement of a binary number is found by adding 1 to its one's complement.

Steps:

- At first, 2's complement of the subtrahend is found.
- Then, it is added to the minuend.
- If the final carry over of sum is 1, it is dropped and result is positive.
- If there is no carry over, the two's complement of sum will be the result and it is negative.

```
e.g.      10110 – 11010
          2's complement of 11010 is (00101+1)
                         i.e. 00110
Hence,            Minuend                  10110
          2's complement of subtrahend + 00110
                                         -------
                                           11100
                                         -------
```

As there is no carry over, the result of subtraction is negative and is obtained by writing the 2's complement of 11100 i.e. (00011 + 1) or 00100.

Hence the difference is -100

(iii) Binary Multiplication

- Binary multiplication is done, similarly the multiplication of decimal number.
  There four rules of binary multiplication

A × B	Multiplication
0 × 0	0
0 × 1	0
1 × 0	0
1 × 1	1

e.g. 1 1 0 1 0 × 1 0

$$\begin{array}{r} 1\,1\,0\,1\,0 \\ \times \quad 1\,0 \\ \hline 0\,0\,0\,0\,0 \\ 1\,1\,0\,1\,0\times \\ \hline 1\,1\,0\,1\,0\,0 \\ \hline \end{array}$$

(iv) Binary Division

It is similar to decimal division. Binary division comprised of other two binary arithmetic operations as multiplication and subtraction.

e.g. 1 1 0 1 0 ÷ 1 0 1

$$\begin{array}{l} 101\overline{)11010}(101 \quad \rightarrow \text{Quotient} \\ \quad\;\; \underline{101} \\ \qquad\;\; 110 \\ \qquad\;\; \underline{101} \\ \qquad\quad\;\; 1 \quad \leftarrow \text{Remainder} \end{array}$$

Octal Arithmetic

(i) Octal Addition

Octal digits range from 0 to 7. Number after 7 is 10. So, in octal number system-

$$\begin{aligned} 7 + 1 &= 10 \\ 17 + 1 &= 20 \\ 77 + 1 &= 100 \text{ and so on.} \end{aligned}$$

e.g. 25 + 4

$$\begin{array}{r} 1\;\; \\ 2\;5 \\ +\quad 4 \\ \hline 3\;1 \\ \hline \end{array}$$

So, $(25)_8 + (4)_8 = (31)_8$

(ii) Octal Subtraction

The subtraction of octal numbers follow the same rules as the subtraction of numbers in other number system. The only variation is in borrowed number. In the decimal system, you borrow a group of $(10)_{10}$. In the binary system, you borrow a group of $(2)_{10}$. In the octal system you borrow a group of $(8)_{10}$.

e.g. $(456)_8 - (173)_8$

$$\begin{array}{r} 4\,5\,6 \\ -\;1\,7\,3 \\ \boxed{\;2\,6\,3} \end{array}$$

Hexadecimal Arithmetic

(i) Hexadecimal Addition

Hexadecimal digits range from 0 to F. Number after 9 is A and letter after F is 10. So, in hexadecimal number system, F + 1 = 10, 1F + 1 = 20, 29 + 1 = 2A and so on.

We look at the hexadecimal number line for reference to addition:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F …

e.g. $(984)_{16} + (27)_{16}$

$$\begin{array}{r} 9\ 8\ 4 \\ +\quad 2\ 7 \\ \hline 9\ A\ B \\ \hline \end{array}$$

So, $(984)_{16} + (27)_{16} = (9AB)_{16}$

(ii) Hexadecimal Subtraction

The subtraction of hexadecimal numbers follow the same rules as the subtraction of numbers in any other number system. The only variation is in borrowed number. In the hexadecimal system, you borrow a group of $(16)_{10}$.

e.g. $(4A6)_{16} - (1B3)_{16}$

$$\begin{array}{r} 4\ A\ 6 \\ -\ \ 1\ B\ 3 \\ \hline 2\ F\ 3 \\ \hline \end{array}$$

●●

CHAPTER-2

PROPOSITIONAL LOGIC

Revision Notes

➢ **Propositional Logic**

- A proposition is a statement that has a value, which can be either true or false but can't anything else. The value of a proposition is called its truth value.
- Propositional logic uses statements which are atomic in nature. These statements are called propositions.
- Propositions have the following properties:

(i) A proposition has a truth value which cannot be anything other than True or False.

(ii) Propositions can be connected using connectives to result in another proposition.

(iii) They are represented by using upper case alphabets.

(iv) States of a proposition can be represented by a Truth table.

➢ **Truth Table**

- It is a complete list of possible values of a proposition.
- In truth table, 0 indicates false and 1 indicates True.

➢ **Connectives**

Atomic proposition can be connected to create a large proposition. They connected by using connectives.

Connectives used in propositional logic are:

(i) Negation (~ OR ′) : It is also known as NOT or complement. Negation denoted as P′ or ~P. It is applicable on a single proposition.

Truth table

A	A′
0	1
1	0

(ii) Conjunction (^) : It is also known as AND and denoted with symbol (^). The result of conjunction is true only when both the atomic propositions are true.

Truth table

A	B	AˆB
0	0	0
0	1	0
1	0	0
1	1	1

(iii) Disjunction (∨) : It is also known as OR and denoted with symbol (∨). The result of disjunction is true if any of the atomic propositions is true.

Truth table

A	B	AˇB
0	0	0
0	1	1
1	0	1
1	1	1

(iv) Conditional (⇒) : It is also known as Implies or If-then, and denoted with symbol (→).

e.g. A : You will do hard work.
B : You will get success.
A → B : If you will do hard work, then you will get success.

(v) Biconditional (↔) : It is also known Equivalence or If and only if or iff. It is denoted with symbol (↔).

e.g. A : She is smart.
B : She is beautiful.
A ↔ B : She is smart if and only if she is beautiful.

Other Important Terminology

- **Well Formed Formula (WFF) :** It refers a simple proposition and compound proposition. A WFF can be determined with the help of following properties:
  (i) If *p* is a propositional variable, then it is a WFF.
  (ii) If α is a WFF, then ¬α is a WFF.
  (iii) If α and β WFF, then
  (αˇβ), (αˆβ), (α↔β) are WFF
  (iv) A string of symbols is a WFF, if and only if it is obtained by many applications of rules (i) to (iii).
- **Tautology :** A compound proposition that is always True.
- **Contradiction :** A compound proposition that is always False.
- **Contingency :** A compound proposition that is neither tautology nor contradiction.
- **Satisfiable :** A compound proposition is satisfiable if there is an assignment of truth values to its variables that satisfies it. When a compound proposition is either tautology or contingency then it is called satisfiable.
- **Unsatisfiable :** When a compound proposition is false for all assignments of truth values, the compound proposition is known as unsatisfiable. When a compound proposition is a contradiction then it is called unsatisfiable.
- **Argument :** Sequence of statements (premises). An argument is valid if it is a tautology otherwise it is fallacy.
- **Syllogism :** A logical process of drawing conclusions from given premises. The premises share a common element called middle term.
- **Premise :** It is a proposition from which a conclusion is drawn.
- **Chain Rule :** It states that two conditions of form P ⇒ Q and Q ⇒ R can be combined to create a new condition P ⇒ R.
- **Modus Ponens :** It states that whenever in a condition, the antecedent is true, the consequent will be true.
- **Converse :** For a condition P ⇒ Q, its converse is Q ⇒ P.
  e.g. Let P – The cat was hungry.
  Q – The cat drank the milk.
  P ⇒ Q – If the cat was hungry then the cat drank the milk.
  Q ⇒ P – If the cat drank the milk then the cat was hungry.
- **Inverse :** For a condition P ⇒ Q, its inverse is P′ ⇒ Q′
  e.g. Let P – The box was locked.
  Q – The box had gold biscuits in it.
  P ⇒ Q – If the box was locked then the box had gold biscuits in it.

P′ ⇒ Q′ – If the box was not locked then the box did not have gold biscuits in it.

- **Contrapositive :** It states that whenever in a condition, the consequent is false, the antecedent is also false. For a condition P ⇒ Q, its contrapositive is Q′ ⇒ P′

  e.g. Let P – The car tyre get punctured.
  Q – The car could not move.

  P ⇒ Q – If the car tyre got punctured then the car could not move.

  Q′ ⇒ P′ – If the car could move then the car tyre did not get punctured.

Know the Terms

- Logic means some rules which can be applied over an expression to find its optimal solution.
- Mathematical logic is used in designing the circuits in the computer system.
- Propositional logic is a declarative statement with two possible values either True or False but not both.

●●

CHAPTER-3

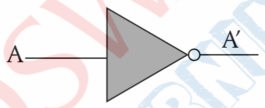
HARDWARE IMPLEMENTATION

Revision Notes

- A logic gate is an elementary digital circuit that gives the result of a propositional operation
- Signals that are used as input and output in a logic gate can be either 0 or 1. Digital circuits are made by using logic gates.

Basic Logic Gates

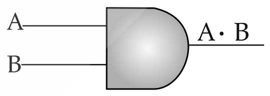
- **NOT Gate:** This gate inverts the input signal to its complement. It implements the operation 'Negation'.



Truth Table

A	A′
0	1
1	0

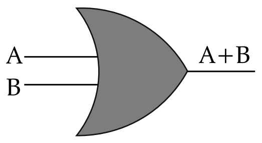
- **AND Gate:** This gate connects two signals using the connective AND. It implements the operation conjunction. It gives a True result only if both inputs are True.



Truth Table

A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

- **OR Gate:** This gate connects two signals using the connective OR. It implements operation `Disjunction'. It gives output true if any of the input signed are true.



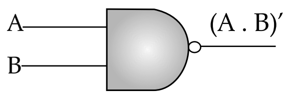
Truth Table

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

Universal Gates

- Universal gates are those which can replace all the other elementary gates.

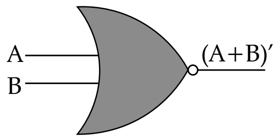
- There are two universal gates – NAND and NOR
  (i) NAND Gate: This gate inverts the signals of an AND gate.



Truth Table

A	B	(A · B)′
0	0	1
0	1	1
1	0	1
1	1	0

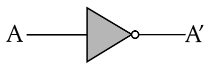
  (ii) NOR Gate: This gate inverts the signals of an OR gate.



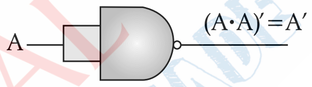
Truth Table

A	B	(A + B)′
0	0	1
0	1	0
1	0	0
1	1	0

- Using NAND and NOR gates, all the three elementary gates can be generated.
- Generating the elementary gates by using Universal NAND gate
  (i) NOT using NAND:

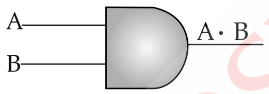


NOT Gate

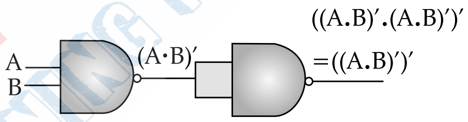


NOT Gate using NAND

  (ii) AND using NAND:

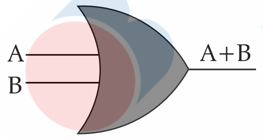


AND Gate

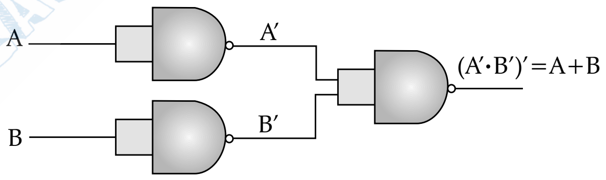


AND Gate using NAND Gate

  (iii) OR using NAND:

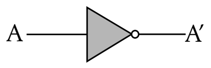


OR Gate

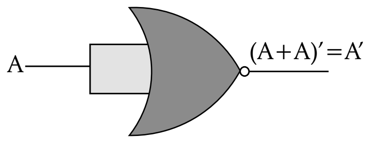


OR Gate using NAND Gate

- Generating the elementary gates by using Universal NOR gates.
  (i) NOT using NOR:

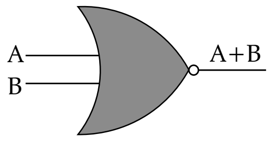


NOT Gate

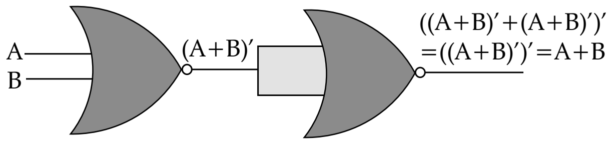


NOT Gate using NOR Gate

  (ii) OR using NOR:

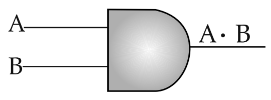


OR Gate

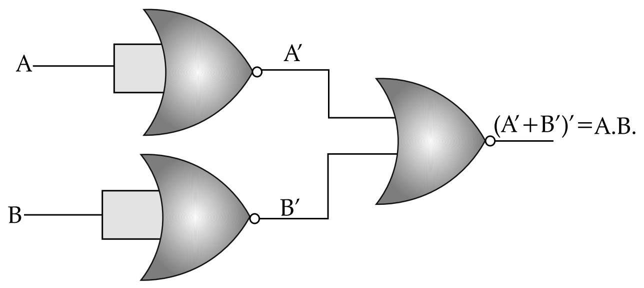


OR Gate using NOR Gate

(iii) AND using NOR:



AND Gate



AND Gate using NOR

Other Special Logic Gates

XOR Gate

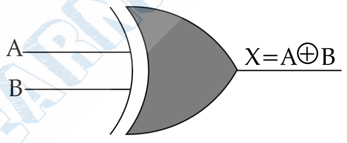
- XOR gate is also known as exclusive OR Gate.
- It is a digital logic gate with two or more inputs and one output that performs exclusive disjunction.
- The output of an XOR gate is True only when exactly one of its inputs is true. If both of an XOR gate's inputs are false, or if both of its inputs are true, the output of the XOR gate is false.

Truth Table for XOR Gate :

A	B	Result
0	0	0
0	1	1
1	0	1
1	1	0

$X = A'B + AB' = A \oplus B$

Logic diagram



XNOR Gate

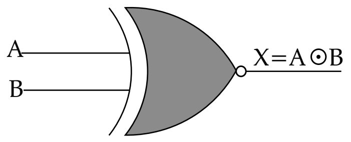
- XNOR gate is also known as exclusive NOR gate.
- It is a digital logic gate with two or more inputs and one output that performs logical equality.
- The output of an XNOR gate is true when all of its inputs are true or when all of its inputs are false. If some of its inputs are true and others are false, then the output of the XNOR gate is false.
- It is denoted by $\odot$ symbol.

Truth Table for XNOR Gate :

A	B	Result
0	0	1
0	1	0
1	0	0
1	1	1

$X = A'B' + AB = A \odot B$

Logic diagram



Sum of Product Form of an Expression (SOP)

- A Boolean expression can be written in the form of Sum of Product terms.
- Product terms are those in which elementary propositions are multiplied. **e.g.** A B′C
- Sum of product terms result as : AB′C + ABC′ +A′BC′+ABC

Product of Sum Form of an Expression (POS)

- A Boolean expression can also be written in form of Product of Sum-terms.
- Sum terms are those in which elementary propositions are added. **e.g.** A+B′+C
- Product of Sum terms result as : (A+B′+C) . (A+B+C′) . (A′+B+C′).(A+B+C)

Half Adder

- A circuit that performs the addition of two bits is called a half adder.
- A half adder circuit is a combinational arithmetic circuit that adds two bits and produces two bits, a sum bit (S) and a carry bit (C) as the output.

  If A and B are the inputs bits, then the sum bit (S) is the XOR of A and B and the carry bit (C) is the AND of A and B.

Truth table of Half Adder

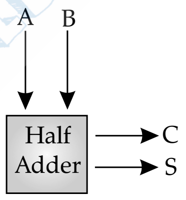
Input		Output	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The simplified expressions of two output lines of a Half adder are :

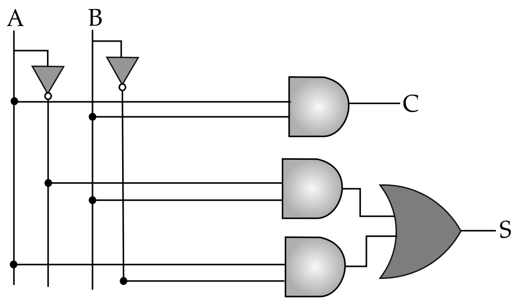
S = A′B + AB′ C = A . B

= A ⊕ B

The block diagram of a Half adder is shown below :



The Logic Gate diagram of an Half Adder :



Full Adder

- A combinational circuit that performs the addition of three bits is called a Full adder.

- A full adder is an arithmetic circuit that adds three bits and produces two bits, a sum bit (S) and a carry bit (Ca) as the output.
  If A, B and C are the input bits then the sum bit (S) is $A \oplus B \oplus C$ and the carry bit (Ca) is A.B + B.C + C.A
  Truth table of a Full Adder :

Inputs			**Outputs**	
A	**B**	**C**	**Carry**	**Sum**
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

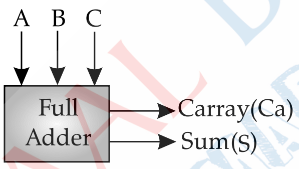
The expressions of the two output lines of a Full Adder :

$$S = \Sigma(1, 2, 4, 7) = A'BC' + A'BC' + AB'C' + ABC$$

$$Ca = \Sigma(3, 5, 6, 7) = A'BC + AB'C + ABC' + ABC$$

The reduced expression of Ca = A.B + B.C + C.A

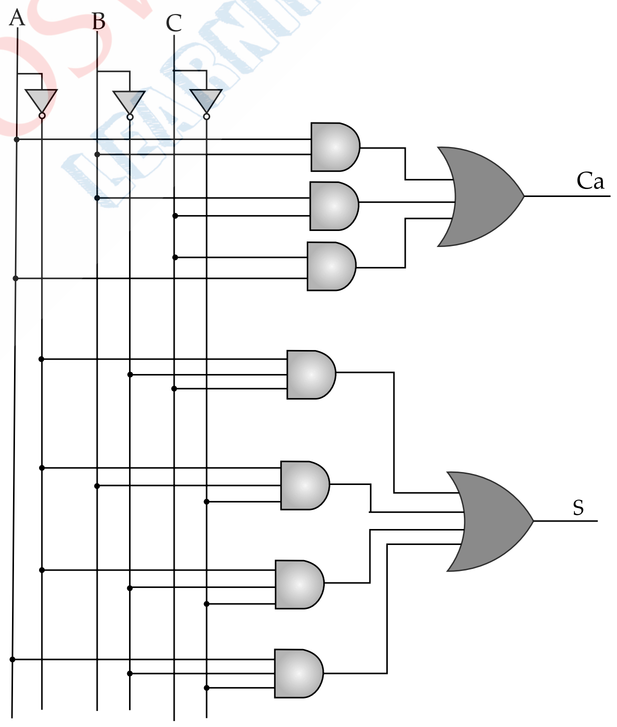
The block diagram of a Full Adder is shown :



The logic gate diagram of a Full Adder :

$$Ca = AB + BC + CA$$

$$S = A'B'C + A'BC' + AB'C' + ABC \quad \text{or} \quad A \oplus B \oplus C$$



CHAPTER-4

JAVA FUNDAMENTALS

TOPIC-1
OOP with Java

Revision Notes

Object Oriented Programming

- In real world, we are surrounded by objects which belong to some class or the other. Object Oriented Programming (also called OOP) applies this concept in programming.
- OOP also implements other concepts which are related to real world.
- Features of Object Oriented Programming are
  - **(i) Encapsulation :** All related data and methods are closed with in one boundary (a class).
  - **(ii) Inheritance :** A derived (child) class is able to access data/methods of its base (Parent) class.
  - **(iii) Dynamic Binding :** Programs are written in such a way that during runtime, the linking of data and methods take place.
  - **(iv) Polymorphism :** Ability of the methods to have the same name but to behave differently under different situations.
  - **(v) Abstraction :** The result of a task is presented without giving the details of how the task is performed.
  - **(vi) Modularity :** A big problem is divided into small working modules which are connectable.

Class and Object

➢ Class
- A class implements the concept of encapsulation. It bounds data and methods that are relevant to one task.
- A class serves as a definition to describe an entity. It is like a blueprint or a template which can be used for creating objects of same type.

  e.g. A house can be seen as a class which encapsulates its necessary member data and methods.

➢ Object
- An object is an instance of a class. A class is a definition of data and methods.
- An object implements that class in reality. So, we can say that in real life, all that we see around us are objects of various classes.

  e.g. The schools that exist in the real world are all objects of class school which have data such as name, students-strength etc and methods such as teaching (), playing (), examination () etc.

Java

➢ **An Object Oriented Programming Language**
- Java was developed by James Gosling, Mike Sheridan and Patrick Naoghton, and sun microsystems in the year 1991.
- This language was initially called Oak. In 1996, it was renamed as Java.

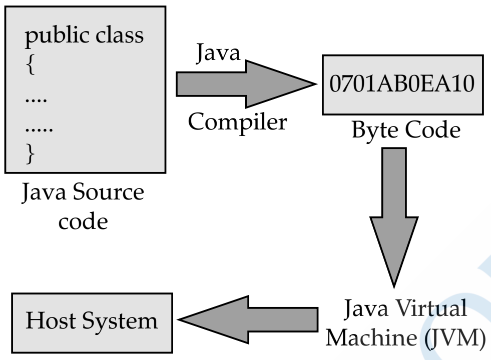
➢ **Features of Java are as follows:**
- **(i) Simple :** Java programming language provides many assisting tools that make it easy to learn.
- **(ii) Platform Independent :** Programs written in Java can be executed on any other platform (such as windows, Linux, etc).
- **(iii) Portable :** The feature of platform independence makes a Java program portable, following concept of "Write Once Run Anywhere (WORA)".
- **(iv) Dynamic and Distributed :** Java programs can use other required files either from the local drive or from other computers by getting connected to the Internet.
- **(v) Robust :** Java provides powerful error (exception) handling mechanisms.

(vi) Secure : Java programming language is secure.

(vii) Object Oriented : Java full fills all important features of object oriented programming.

Java Byte Code

- Java code is compiled to create Java Byte Code which is then processed by Java Virtual Machine (JVM). This code is independent of the processor of the computer system.
- The portability or machine independence of Java program is because of this Java Byte Code.



Java Virtal Machine (JVM)

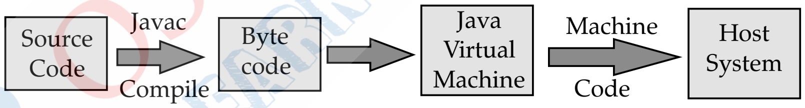
- This is a software that convert Java Byte Code to Machine Level Language. It works as a platform for a Java program and can be executed on any platform (Windows, Linux etc).
- JVM comes is between a compiled Java code and the computer system.
- JVM is the main component of the Java architecture and is a part of the Java Runtime Environment (JRE). JVM combined with Java API (Application Program Interface) make the Java platform.
- Most compilers provide code for a computer system but Java compiler produces code for JVM.

➢ **The functioning of Java Virtual Machine as**

(i) When a Java program is compiled, the compiler creates a code called Java Byte code for the JVM.

(ii) Now, for the different operating systems, this Java Virtal Machine converts the Java Byte to a machine executable code.

(iii) Hence, if the operating system is windows, JVM creates a machine code that is executable in it. A different machine code is created by the JVM if the operating system of the computer is Unix or Linux and so on.



Java Development tool Kit (JDK)

➢ The JDK additionally contains the development tools necessary to create Java programs. It consists of a Java compiler, the Java virtual machine and the Java class libraries.

BlueJ

➢ BlueJ is a development environment that allows you to develop Java programs quickly and easily. Its main features are.

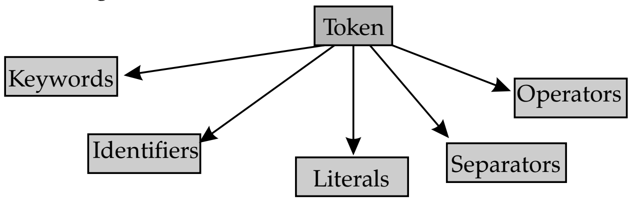
- Simple
- Designed for teaching
- Interactive
- Portable
- Mature
- Innovative

TOPIC-2
Java Fundamental

Revision Notes

Token

- A token is the smallest unidentifiable part of a program.
- Following are the tokens in Java



(i) Keywords

- These are the words that convey a specific meaning to the compiler. These are reserved words and are used for special purposes and thus, cannot be used as normal identifiers.
- List of keywords used in Java :

abstract	char	else	if	native	short	throw
boolean	const	extends	implements	new	static	throws
break	continue	final	import	package	strictfp	transient
byte	double	finally	instance of	private	super	try
case	default	float	int	protected	switch	void
catch	do	for	interface	public	synchronised	volatile
class	enum	goto	long	return	this	while

(ii) Identifiers

- Identifiers are names given to variable, constants, classes, methods.
- Keywords cannot be used as identifiers.
- The first character of an identifier must be letter, an underscore (_) or a doller sign ($).
- Identifiers are case sensitive. This means that hello and Hello are different.

➢ **Valid identifiers**

Subvalue, Num, totmark, sum123

➢ **Invalid identifiers**

Sab value , A + B2, int, 2 Num

(iii) Literals

Literals are constants that have a fixed value of any of the data type.

e.g. 9, 3, 7 are numeric literals

'a' is a character literals

'True' is a Boolean literal

(iv) Separators

Separators help to define the structure of a program. A separator is one or two tokens that separate some language features from other features.

Some separators are () , {}, [] , . , , , ; etc.

; is also called statement terminator.

(v) Operators

- These are basically the symbols or tokens which perform arithmetical or logical operations to yield results.

- They are used to manipulate primitive data types.

Operators are following types :

(a) Unary operators : These operators are used with one variable to indicate its sign, such as ++, – – etc.

(b) Binary operators : These are used to perform operation between two operands.

Arithmetic operators	Relationship operators
+ Addition	= = equal to
– Subtraction	! = Not equal to
* Multiplication	> Greater than
/ Division	< Less than
% Modulus	>= Greater than or equal to
	<= Less than or equal to

Logical operators	Bitwise operator
&& Logical AND	& Binary AND
|| Logical OR	| Binary OR
! Logical NOT	^ Binary XOR

➢ **Conditional operator (?:)**

This operator is used to found the condition result, which could be either true or false.

Syntax

Variable x = (expression) ? value 1 (if true) : value 2 (if false);

Operator Precedence

It shows the order of operator in priority from top to bottom in different rows :

	Operator	Associativity	Precedence
() [] . ->	Function call Array subscript Dot (Member of structure) Arrow (Member of structure)	Left-to-Right	Highest 14
! ~ – ++ –– & * (τψπε) σιζεοφ	Logical NOT One's-complement Unary minus (Negation) Increment Decrement Address-of Indirection Cast size in bytes	Right-to-Left	13
* / %	Multiplication Division Modulus (Remainder)	Left-to-Right	12
+ –	Addition Subtraction	Left-to-Right	11
<< >>	Left-shift Right-shift	Left-to-Right	10
< <= > >=	Less than Less than or equal to Greater than Greater than or equal to	Left-to-Right	9
== !=	Equal to Not equal to	Left-to-Right	8
&	Bitwise AND	Left-to-Right	7
^	Bitwise XOR	Left-to-Right	6
|	Bitwise OR	Left-to-Right	5
&&	Logical AND	Left-to-Right	4

||	Logical OR	Left-to-Right	3
?:	Conditional	Left-to-Right	2
=,+= *=,etc.	Assignment operators	Left-to-Right	1
,	Comma	Left-to-Right	Lowest 0

Table : Precedence and Associativity Table

➢ **Statement**

It is the smallest unit of a program that gets executed. Statements in Java are terminated by a semi-colon (;).

➢ **Block**

- A group of statements that have a specific purpose of called a block. They are used to divide big tasks into smaller ones.
- It helps in understanding bigger problems.
- Blocks are generally enclosed within a pair of parenthesis ({}).

➢ **Comments**

- Comments are those parts of a program that do not get executed. They are used to variate notes for better understanding of a program.

Java has two types of comments as

- Single line comment indicates by //
- Multi-line comment indicates by /* and */.

➢ **Data Types**

- It refers to the type of data that can be stored in a variable.
- Sometimes, Java is called strongly typed language because when you declare a variable, you must specify the variable type.

Data types in Java are classified into two types :

(i) **Primitive Data Types :** These are the data types defined by the language itself. Java supports 8 primitive data types, they are predefined and all data types have predefined keywords.

Primitive data types are as follows :

Type	Size	Example
byte	1 byte	byte b =2;
short	2 bytes	short s = 7;
int	4 bytes	short a = 5;
long	8 bytes	long n = 980807465;
float	4 bytes	float d = 45.7;
double	8 bytes	double d = 72.88934445634;
char	2 bytes	char h = 'd';
boolean	1 byte	boolean st = true;

(ii) **Non-Primitive Data Types :** These are referenced data types which are based on primitive data types. Non-primitive data types store the memory address of an object. Class, interface and array are the examples of referenced data types.

➢ **Variable**

- It is a container that holds values that are used in a Java programming language. It can possess any combination of letter without space.
- Each variable in Java has a specific type, which determines the size and layout of a variable's memory.
  - **(i)** Specify the name of the variable
  - **(ii)** Specify the data types of the variable
  - **(iii)** Specify the scope of usage of the variable. Syntax: data type variable-name;

    e.g. int age;

➢ **Variable Types :** Java programming language defines the following kinds of variable:

- **Instance variables** are used to store the state of an object. Every object created from a class definition would have its own copy of variable.

  Syntax : object Reference. Variable Name

- **Class variables** are created when the program starts and destroyed when the program stops. It can be accessed by calling with the class name. Class variables also known as static variables are declared with the static keyword in a class, but outside a method, constructor or a block.

  Syntax : class name. variable name
- **Local variables** are declared in method, constructors or blocks. These are implemented at stack level internally.

  Syntax : variable name

➢ **Expression**

- It is a construct mode of variables, operators and method invocation, which are raised according to the syntax of the language, that evaluates to a single value.

e.g.

$a = 4.23 - 2x^2$

$x+y$

In Java , syntax of the above expression will be

$a = (4.23 - 2^* \; x \; ^* \; x) \, / \, (x + y);$

Type Casting

- When data of multiple types are operated upon, the result varies if the data types are not correctly set.
- The compiler has its own rules in cases when an expression uses data of multiple types. When data of multiple types are operated upon, the result varies if the data types are not correctly set.
- The compiler has its own rules in cases when an expression uses data of multiple types.
- Type casting is nothing but assigning a value of one primitive data type to another. When you assign the value of one data type to another, you should be aware of the compatibility of the data type. If they are compatible, then Java will perform the conversion automatically known as Automatic Type Conversion and if not, then they need to be casted or converted explicitly.
- In java, here are two ways of type casting.

(i) Explicit Type Conversion: In this kind of type conversions the resulting data types are explicitly specified by the programs. It is also known as Narrowing Casting.

$$\underset{\xrightarrow[\text{Narrowing}]{\hspace{12em}}}{\text{double} \to \text{float} \to \text{long} \to \text{int} \to \text{short} \to \text{byte}}$$

e.g. double x = 10;

int = (int) x ; //explicit type casting

(ii) Implicit Type Conversion: In this kind of Type Conversion, the resulting data types are not specified and are chosen by the compiler. The compiler prefers not to lose any past of the data value. It is also known as Widening Casting.

$$\underset{\xrightarrow[\text{widening}]{\hspace{12em}}}{\text{byte} \to \text{short} \to \text{long} \to \text{float} \to \text{byte}}$$

e.g.

```
byte   b = 10 ;
int     i = b ; // implicit type casting
long    i = i ;  // implicit type casting
```

➢ **Wrapper class**

- A variable of primitive data type is by default call by value. To achieve call by reference, we use the wrapper class.
- Wrapper class is used to convert any primitive data type into object.
- A wrapper class also contains a number of other different methods, which may be used in the processing of the corresponding data type.

Following table shows data type with its wrapper class:

Data Type	Wrapper Class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

char	Character
boolean	Boolean

●●

CHAPTER-5

JAVA STATEMENTS AND SCOPE

Revision Notes

- Statements in Java are syntactically complete instructions that perform a specific task. A program is hence, made up of multiple statements that are arranged in a, way to fulfil a given aim.
- A Statement can have multiple inner components or expressions.
- A Java program contains classes and these classes have some methods which contain statements that are executed by the compiles.

Selection or conditional statements

These statements are used in a program to choose different paths of execution based upon the outcome of an expression or the state of a variable. It is also called decision state.

These are following conditional statements in Java:

(i) If Statement : It executes a single block of statements if the specified condition evaluates to "True" otherwise the given set of statements are ignored.

Syntax

```
if (condition)
{
      statement(s) ;
}
```

(ii) If...then....else statements : In this a single statement or a group of statements enclosed within braces. The if block is executed only when if condition is "True", otherwise, the "Else" block is executed.

Syntax

```
if (condition)
{
       statement 1 ;
}
else
{
       statement 2 ;
}
```

(iii) If...else....if statements : In this, a number of logical conditions are checked for executing various statements. It is based upon nested if - else and is often called the if – else – if ladder.

Syntax

```
If (condition 1)
{
    Statement 1;
}
Else if (condition 2)
{
    Statement 2;
}
else if (condition 3)
{
    statement 3;
}
else
{
    statement (s);
}
```

If the condition 1 is true, then statements 1 gets executed otherwise if condition 2 is true then statement 2 gets executed and so on. If none of the conditions are true then the last statement gets executed.

(iv) Switch case statements : It works as a multi way branch, statement. In this statement, if none of the cases match, then default becomes active. Default block present under it should appear at last. It is optional.

Syntax

```
switch (expression)
{
    case value 1 :   statements ;
                break ;
    Case value 2 :   statements ;
                break ;
    Case value 3 :   statements ;
                break ;
    default : final block of code ;
}
```

Fall through in switch case

Break statement is very important in switch case construct. It helps to terminate a block and to bring the central out.

If break is not given then, all the statements that appear below the matching case, will execute until a break statement is reached. This leads to a logical error situation. Which is called fall through.

Looping Statements

A loop is a programming concept in which a group of statements are executed repeatedly as long as the specified condition is valid.

These are three types of loops in Java as follows :

(i) For loop : It is also an entry controlled loop. It is a short method of looping which is commonly preferred by programmers. This loop tests the condition before entering the loop body, if the condition evaluates to "True" the given statements are executed else the loop will not execute.

Syntax

```
for (initialization ; condition ; increment/decrement)
{
       block of statements
}
```

(ii) while loop : It is also known as entry controlled loop. This loop checks the condition first before entering the loop. If condition evaluates to true, the statements given inside the loop get executed.

Syntax

```
while (condition)
{
       block of statements
       increment/decrement;
}
```

(iii) do – while loop : It is known as exit controlled loop. This loop ensures that the complete loop is executed at least once, then it checks whether the given condition is true or false. If the condition evaluates to "True", then the loop will continue but if the condition is "False" the loop will be terminated.

Syntax

```
do
{
```

```
                    Block of statements
                    increment/decrement;
               }  while (condition) ;
```

Block of code

- A group of program statements that are enclosed within a pair of curly { } braces is called block of code.
- When one block is present inside another block is called nesting of blocks.

Scope of a variable in a block

- If a variable is declared inside a block then its existence is limited to the block.
- Two variable of the same name can exist if they are declared in different blocks, the variable declared in the linear block gets priority.

Jump statements

- These are used to transfer control from one point in the program to another point, at any time. These statements can be used to modify the behaviour of conditional and iterative statements.
- Jump statements allow you to exit a loop, starts the next iteration of a loop or explicitly transfer the program control to a specified location in your program.

(i) **break :** It causes a loop to terminate its iterations. Whenever a break is encountered, the control comes out of the loop and resumes at the next statements following the loop.

(ii) **Continue :** It makes the loop-counter go to its next value by skipping the rest of the statements for that particular iteration.

Know the Terms

- A switch as part of the statements sequence of an outer switch is called a **nested switch** .
- The most basic method of looping in Java is **iteration**.
- **Return statements** is used to explicitly return the expression from a method.
- **A label** is any valid Java variable name.
- **When a break statement is used inside a nested loop, then only the inner loop gets terminated.**

●●

CHAPTER-6

EXCEPTION HANDLING

Revision Notes

Errors and Exceptions

- Error is a common term that indicates that something is wrong somewhere.
- In Java, errors are of a series concern and therefore are handled with much care. They are classified into Errors and Exceptions.
- An exception is an erroneous situation in a program which causes the program to end abruptly. A program is called robust if it does not crash during execution.

Types of Errors

1. Compile time error [syntax error]
2. Logical error
3. Runtime error [Exceptions]

- In java programming, importance is given to a robust error free program. Exception handling therefore, exists to handle those situation that may cause a program to crash.

Compile Time Errors (Syntax Errors)

These are caused by an improper way to writing a program and so, these are the most common errors with the beginners.

The good thing about syntax errors is that the compiler can detect them. The compiler also generates a message about the error which can be taken as a clue to rectify it

Some of the most common syntax errors are

- **Wrong spelling of keywords :** One has to be very careful while writing long class names where both upper and lower case characters are used.
- **Missing semicolon :** Statement terminator indicated by a; must be present to mark the end of any command.
- **Data type mismatch :** Care must be taken towards verifying the data types of the variables, when more than one are involved.
- **Unclosed character or string :** Caused when the compiler encounters any character (‘ ’) or string (“ ”) not closed.
- **Method prototype and method call :** A method prototype should be given a proper return type, acceptable method name, parameter list along with their data types.
- **Accessing a variable out of its scope :** A variable is bounded within its block.
- **Array index out of bounds :** In case of arrays, the cell numbers begin with 0. In case of string handling also, character positions are numbered from 0.

Logical Errors

These are caused due to wrong interpretation of the logic. They are more difficult to be identified as the compiler does not list them.

The programmer has to apply careful strategies to debug a program from logical errors.

Hints to identify logical errors :

- Counter initialization.
- Use of break/continue in a loop upon condition.
- Reinitializing variables inside a nested loop.
- Fall through in switch case construct.
- Loop ranges, condition validity.
- In case of while and do… while loops, modification of the loop counter or terminating criteria should be given more attention.
- It has been observed that logical errors can be identified with intermediate outputs of the variables that are used in the process.

Runtime Errors

These are not caused by the programmer’s fault, but due to external reasons. Java provides different ways to catch these errors by a series of exception handling statements.

Some logical errors cause runtime errors. If runtime errors are not handled carefully, they can cause the program.

Some of the main causes of exceptions are

- Stack overflow during some huge mathematical calculations.
- Erroneous input.
- Calculation causing division by zero.
- Resources unavailable.
- Too many processes loaded into the system, causing shortage in memory availability.

Handling Exceptions in Java

Java provides a class called the Exception class that comprises of the various error handling subclasses and methods like, the try catch block, the keywords-throw, throws, finally etc.

In a method, an exception can be identified and treated with the help of a try catch block.

The probable error causing code is enclosed in a try catch block. In case any error is generated, then it is caught by the catch block.

Try & Catch Block

Try block : It contains a block of program statements within which an exception might occur. A try block is always followed by a catch block, which handles the exception that occurs in the associated try block. A try block must be followed by a catch block or finally block or both.

Syntax

```
try
{
    statements that may cause an exception
}
```

Catch block : It must be associated with a try block. The corresponding catch block executes if an exception of a particular type occurs with the try block.

Syntax

```
try
{
    statements that may cause an exception
}
    Catch (exception (type) e (object))
{
    Error handling code
}
```

Exception Object

In the try catch block, an exception object is created for the exception that occurs. This object can be accessed inside the catch block.

Multiple catch blocks

In case of multiple catch blocks, the control searches for the matching exception and that specific catch block gets executed and other catch blocks remain inactive.

Finally keyword

The keyword finally is used in a try catch block which becomes active after all the try blocks have been executed. The finally block is optional.

It gets executed irrespective of whether any exception was thrown or not. The main use of finally block is to do cleanup tasks, since it gets executed in all cases.

Keywords Throws and Throw

Throws : The keyword throws is used by the compiler to generate a predefined error. For example, IO Exception (Input Output Exception) is generated when an alphabet is imputed in a variable of integer type.

"throws" keyword is written in the method prototype.

Throw : There is another keyword throw that allows the programmer to create a new error situation that is specific to the program problem.

For example, in a banking program, amount less than a certain value in an account might be termed as Minimum Balance.

Exception class

The exception class is a subclass of the Throwable class. Questions other than the exception class there is another subclass called Error which is derived from the Throwable class. The Exception class has two main subclasses :

IO Exception class and Runtime Exception class.

●●

CHAPTER-7

METHODS (FUNCTIONS)

Revision Notes

- A method or a function is a named unit that has an aim and contains a single/multiple line code.

- Methods make a program easy to understand. They help in code reusability. They also implement code abstraction by separating complex code.

Function Prototype

It is the way of defining a function. The prototype is the first line of the function.

Syntax

Return_type   function_name (Argument list)

Here,

Return_type: A function can return only one value, which may be a primitive of any type or an object. No return is denoted by void.

Function_name: Any identifier can be used to name a function. A function is addressed by its name followed with (). e.g. Add ()

Argument/Parameter list: A function can accept one or more arguments which can be either a primitive or an object.

Method call

- Methods cannot run on their own, they need to be invoked. Methods will be invoked or called by the objects of the class.
- It can be called by providing the method name, followed by the arguments being sent enclosed in parenthesis.

There are two ways to call a method:

(i) Call by value

It is also referred as pass by value that is, the value held in the variable that is passed as an argument is copied into the parameters that are defined in the method prototype.

That is why changes made to the variable within the method had no effect on the variable that was passed.

(ii) Call by Reference

It is also known as pass by reference. This method is valid for passing an object reference to any method call. When we declare a reference type variable, the compiler allocates only space where the memory address of the object can be stored.

The space for the object is allocated at the time of object creation using the new keyword. A variable of reference type differs from a variable of a primitive type in the way that a primitive type variable holds the actual data while a reference type holds the address of the object, not the data.

Returning from a method

- We can specify return type of the method as primitive data type of class name.
- Return type can be void means it does not return any value.
- Method can return a value by using return keyword.
- The data type of the return value must match the method's declared return type, you cannot return a float value from a method declared to return an integer.

Super and Sub function

In a function call, the calling function is called the super function and the called function is called the Sub function. In other words, SUPER function calls a SUB function.

Function and Class

A class may contain one or multiple functions called member functions. Program code can be written only in a function of a class.

Local data of a Function: Data that are declared in a function are called local data of that function. Scope of these local data is within that particular function.

Member data of a class: Data which are declared in a class, outside all the member functions are called member data of that class. Scope of this member data is within that particular class. They can be accessed by all the members of that class. Any function can change a member data. The effect of the change can be seen from any other function of the class.

Importance of main ( ) method

Every program should contain a controlling method called main ( ). Conventionally program execution begins from the main ( ) method.

Actual and Formal Parameter

The parameters that are used in the Super function, while calling the Sub functions are called Actual parameters. The parameters that are used in the Sub function are called Formal parameters.

Nesting of Function

This concept is implemented when one function makes a call to another function.

e.g. fnTotalFactors ( ) is the sub-function and fnIsPrime( ) is the super function. Also, fnIsPrime( ) is the sub function of main ( ).

'this' keyword

'this' keyword is used in a method to refer to the object that invoked it, i.e. it refers to the current object. Whenever a reference to an object of the current class type is required, 'this' can be used.

It can also differentiate between instance variables and local variables. The exact purpose of this is to remove ambiguity between local and instance variables.

Pure and Impure functions

A function is called pure function if it always returns the same result for same argument values and it has no side effects like modifying an argument (or global variable) or outputting something. The only result of calling a pure function is the return value. Examples of pure functions are strlen(), pow(), sqrt() etc.

Example of pure functions:

- Display the value of arguments
- Print the multiplication table of a number sent as an argument.
- Return the sum of two arguments sent as arguments.

Impure functions, also called modifier functions, are those that can cause a change of state in the object. That means, values of the object's instance variables get modified or changed depending on the current state of the object on which the function operates.

Example of impure functions:

- Increase the value of its argument by 1, each time the function is called.
- Modify the length of a ribbon by taking the length and the amount of change.

Mechanisms for Side Effects

A side effect method is a method which modifies some state variable value/arguments passed having a consequence beyond its scope, that is to say it has an observable effect besides returning a value (the main effect) to the invoker of the operation. In simpler terms, a method can take arguments and run some logic and return a value or not return anything.

Now if it does not return a value then it must be a side effect method.

'static' keyword

A class can have multiple objects, where the object have their own copy of the class members. But some members of the class are unique which are not meant to be accessible to the objects. Those members are called static members. They are common to all objects.

Static members of a class are called class Members and non-static members of a class are called Instance Members. Non static variables cannot be used in a static method.

Constructor

A constructor is a member function of a class. It has some special features :

- A constructor has the same name as that of the class.
- It cannot be called as other functions. It gets automatically executed whenever an object of that class is created.
- It does not have any return type, not even void.
- It may or may not take arguments.

A constructor is used to initialize the member data of a class to their initial values. In general, the member data are initialized to null.

Default Constructor: When a program does not contain any constructor, java compiler provides a default constructor. This constructor cannot be seen in the program. It initializes the member data variables to their respective null values and it gets activated only when the program does not contain any constructor.

No-argument constructor : Constructor with no arguments is known as no-arg constructor. The signature is same as default constructor, however body can have any code unlike default constructor where the body of the constructor is empty.

Parameterized constructor

It is a constructor having a list of arguments (or parameters). Since a constructor gets executed automatically at the time of object creation, the values of these parameters must be supplied during object creation.

CHAPTER-8

ARRAYS

Revision Notes

- An array is a collection of multiple variables which share the same name and data type. It is used in cases where large number of similar type of data has to be used.
- Suppose, we want to store the marks obtained by 50 students in a subject. We can either use 50 variables or we can use one array of size 50 (data type of the array being integer or double).
- Each array has a name, type and size. It can store multiple values depending on its size. All the values have to be of the same type. The array stores the values in cells. The cells can be addressed by using the cell number.

  Syntax

  ```
  Data_type Array_name[] = new Data_type [size];
  e.g. int arr[] = new int[5];
  ```

  Here,

  (i) The brackets [ ] in the declaration statement indicates the size of the array.

  (ii) In the working code, the [ ] brackets indicate the cell numbers, or index.

  (iii) The cell numbers begin with 0.

  (iv) If the array size is n then cell numbers range from 0 to (n – 1).

arr [5]	30	45	15	76	42
Index	0	1	2	3	4

  We can say.

  ```
  arr [0] = 30; arr [1] = 45; arr[2] = 15; arr [3] = 76; arr [4] =42;
  ```

- **Dynamic and Static Array Allocation**
  - ➢ An array is a collection of variables, where each variable is present in each cell of the array.
  - ➢ The cell numbers are continuous integers and hence can be generated by using a loop. It is called dynamic allocation. In dynamic allocation, even the size of the array can be a variable.
  - ➢ An array can be declared along with its values.

  Syntax

  ```
  int ar[] ={10,20,30,40,50};
  ```

  - ➢ In this case, the array gets declared and filled simultaneously. This is called static allocation.
- **Use of Keyword 'new'**
  - ➢ Keyword new is used to allocate memory. In case of an array, memory requirement varies with the size of the array. So, keyword new is used to allocate memory to an array depending on its requirement.
- **Two Dimensional Arrays**
  - ➢ A two dimensional array is a collection of elements where the elements are logically arranged in rows and columns. A 2D array is used for handling data arranged in tabular form. It can work with more data in a 1-D array.
  - ➢ A 2D array, Arr [3] [4] as shown below:

- The name of the array here is Arr. It has 3 rows and 4 columns.

  Syntax

```
    Data_type array_name[][] = new Data_type[row_size][column_size];
e.g.
    int Arr[][]= new int[3][4];
```

- The row and column numbers begin with 0. Each element is located with the help of its row and column value. Let us assume another example of a 2D array containing random integers.

Arr[3][4]	0	1	2	3
0	10	5	8	2
1	15	25	28	6
2	7	12	18	16

Note: Elements of a 2D array are accessed using a nested loop of order 2.

Know the Terms

- An array offers a convenient means of grouping related information.
- Searching an element in an array
  (i) Linear Search
  (ii) Binary Search
- Sorting an array
  (i) Bubble sort
  (ii) Inserting sort
  (iii) Selection sort

CHAPTER-9

ARRAY APPLICATIONS SORT AND SEARCH

Revision Notes

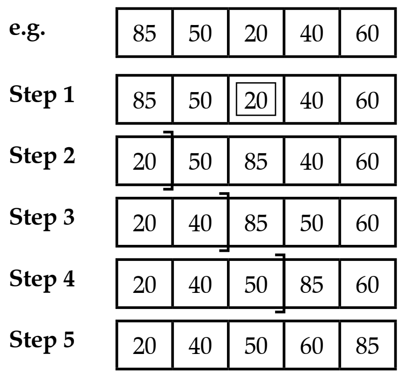
Sorting

- Sorting is defined as arranging data in a certain order, is a very common activity in data processing.
- Sorting of an array means arranging the array elements in a specified order i.e. either in ascending order or descending order.

  There are different types of sorting techniques available to sort a group of elements as
  (i) Selection sort
  (ii) Bubble sort
  (iii) Insertion sort

  Selection Sort : The basic idea of a selection sort is to repeatedly select the smallest or largest key in the remaining unsorted array and placing it at the first position of the array. Then, the process is repeated with rest of the array, *i.e.* the next smallest / largest element is selected and put onto the next slot and so on till the last of elements.

  The principal of selection sort for arranging an unsorted list in ascending order is:
  - Select the smallest element in the unsorted list of size n.
  - Swap it with the first element of the unsorted list
  - Modify the unsorted list by moving the initial or the or the starting element.
  - Repeat the steps for (n-1) times

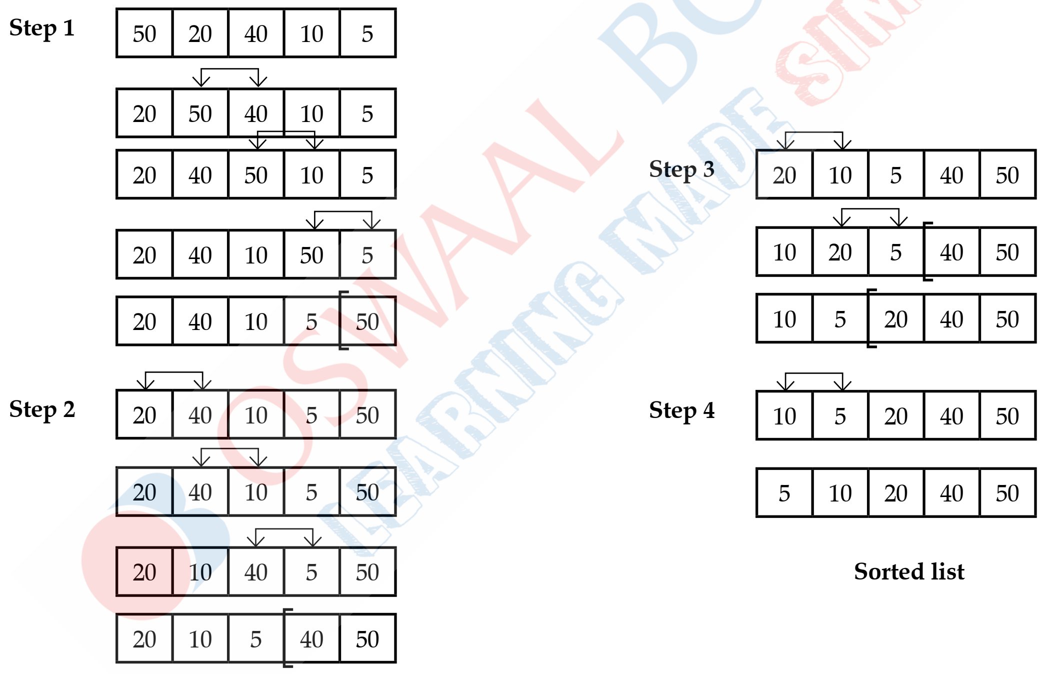


Bubble Sort : The basic idea of bubble sort is to compare two adjoining values and exchange them, if they are not in proper order.

The principal of bubble sort for arranging an unsorted list in ascending order is:

- Compare two consecutive elements and swap if required.
- The largest unsorted element gets its place after each complete traversal.
- Repeat the process (n-1) number of times.
- Modify the unsorted list by reducing the size of the list by each time.

e.g. list is 50 20 40 10 5

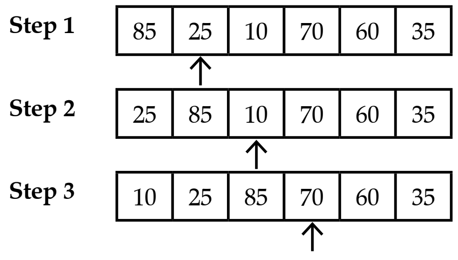


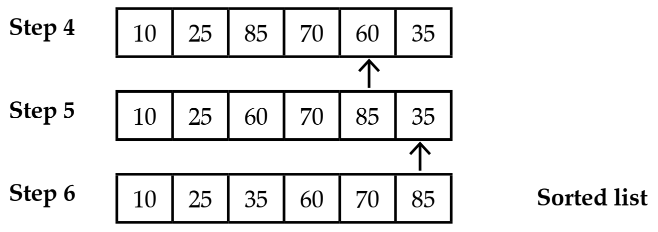
Insertion sort : In this sort, initially the first element is picked up in the unsorted part and then appropriately inserted in the sorted part, this process will repeat till the final list is ordered accordingly.

The principal of Insertion sort for arranging an unordered list in ascending order is:

- Pick an element and find its suitable position on its left.
- Insert the element by right shifting the other elements by one place.
- Repeat the process for elements in 2nd to last position.

e.g. list is 85 25 10 70 60 35





Searching

It involves searching for the specific data element in an array. In general, the process of search results in a Boolean value of true or false.

Searching can be done in two ways:

(i) Linear search

(ii) Binary search

Linear Search : This process is simple. Compare all the elements one by one with the element that is being searched.

- Run a loop from the beginning to the end of the array.
- Compare the element being searched with the current element.
- If a match is found, then the element is present. Terminate the loop.
- If no match is found till the end of the loop. The element is not present.

Binary Search : This process works only on presorted arrays. It is a time saving algorithm for very long list of values. It works by halving the arrays with each failed search comparison.

- Assume the lower limit to be low and the upper limit to be high.
- Find the middle position of low and high, say mid.
- Compare the search elements with the element at position mid.
- If a match is found then the process terminates.
- If a match is not found then –

(i) Change low to (mid +1) if the element being searched is larger than the element in mid.

(ii) Change high to (mid -1) if the element being searched is smaller than the element in mid.

●●

CHAPTER-10

STRINGS

Revision Notes

○ All the keyboard characters fall under the primitive category char. A char value can only one character in it and they are enclosed within single quotes ' '.

e.g.
```
char ch = 'A'
char ch = '#';
```

○ A group of characters form a string. Java provides us with a class called string that helps to create and manipulate strings. The string class in Java also has a set of library methods.

e.g.
```
string sts = "Hello word";
```

○ **Features of Java characters and strings**

(i) characters fall under the primitive category char. A string is a java class that creates a word or sentence using a group of characters.

(ii) A character is enclosed within single quotes '#'. A string is enclosed within double quotes "#".

(iii) characters have a fixed size of 2 bytes. Size of string is not fixed.

(iv) Java has a wrapper class called character that provides character handling methods.

○ String is not a primitive class but it is Java class that provides string handling methods.

○ **String Library Methods**

Method	Detail
int length ( )	It returns the length of the string.

char charAt (int p)	It returns the character present at position p of the string.
int indexOf (char ch)	It returns the first index of character ch in the string.
int lastIndexOf (char ch)	It returns the last index of character ch in the string.
int compareTo (String s2)	It returns the numeric difference between the characters of the current string and the argument string s2. It uses the ASCII value and subtracts the 2nd string from the 1st string.
boolean endswith (String s2)	It returns the boolean result of comparison of the argument string s2 present at the end of the 1st string. It is case sensitive.
boolean equals (String s2)	It returns the boolean result of comparison of equality between the current string and the argument string s2. It is case sensitive.
string concat (String s2)	It concatenates the argument string s2 behind the current string and returns the new string.
boolean startswith (String s2)	It returns the Boolean result of comparison of the argument string of the 1st is case sensitive.
String replace (char oldchar, char new char)	It returns a new string using the a current string in which all the occurrences of a oldChar character is replaced with newChar character.
String substring (int beginIndex, int endIndex)	It returns a new string using the current string whose starting and ending character index number are given.
String toLowercase( )	It converts the current string to lower case and returns it.
String toUppercase( )	It converts the current string to upper case and returns it.
String trim ( )	It trims the spaces present before and after the 1st and last character of the current string and returns it.

- **Character wrapper class**
- The methods of this class perform operations on characters. Some methods either returns a boolean result or any character. The methods names indicate their operation.
  - boolean isDigit (char ch)
  - boolean isLetter (char ch)
  - boolean isLetterOrDigit (char ch)
  - boolean isLowerCase (char ch)
  - boolean isUpperCase (char ch)
  - Char toUpperCase(char ch)
- **Array of Strings**
  - An array of strings contain a list of strings. Each element of the array possesses properties of a string.

Know the Terms

- Math class is present under the package Java lang.
- **String length :** the length ( ) method, which returns the number of characters contained in the string object.

●●

CHAPTER-11

DATA FILE HANDLING

Revision Notes

- Data is one of the most integral part of a program. Data has to be read and data has to be written by a program for the purpose of doing any kind of task.

- There are some cases where the data to be read and written need not be preserved, they are temporarily taken from the user and is lost after the program execution is over.
- On the other hand, there are some cases where the data to be used, has to be preserved for a longer period of time. Also, there are some cases where the program output is very long and do not fit in the console. It is then when data files are required.
- Data files are text files that are placed in the computer memory and used to store data. A Java program can read data from a data file and can write data in a data file.

Scanner Class

The Scanner class is a class which allows the user to read values of various types. Scanner class is available in java. util system package. You must import java.util package to avail the facilities contained in Scanner class.

The statement to import java.util package is shown below :

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

After importing java.util package, create object which can hold a set of values of different types.

The syntax to create a Scanner object is shown below :

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

Methods do Read Tokens from Scanner object

• int nextInt ()	–	Ruturns the next token as an int
• long nextLong ()	–	It returns the next token as a long
• float nextFloat ()	–	It returns the next token as a float
• double nextDouble ()	–	It returns the next token as a double
• String next ()	–	It finds and returns the next complete token from the Scanner object as a string
• String nextLine ()	–	It returns the rest part of the current line, excluding any line separator at the end.
• void close ()	–	It closes the Scanner.

Methods for Checking Appearance of a Specific Token

• boolean hasNextInt ()	–	It returns true if the next token can be interpreted as an int value.
• boolean hasNextLong ()	–	It returns true if the next token can be interpreted as a long value.
• boolean hasNextFloat ()	–	It returns true if the next token can be interpreted as a float value.
• boolean hasNextDouble ()	–	It returns true if the next token can be interpreted as a double value.
• boolean hasNext ()	–	It returns true if the another token of any type is available as its input.
• boolean hasNextLine ()	–	It returns true if the another token of any type is available as its input.

Tokenizer Class

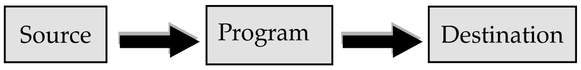
The java.io.StreamTokenizer class takes an input stream and parses it into <tokens>, allowing the tokens to be read one at a time. The stream tokenizer can recognize identifiers, numbers, quoted strings and various comment styles.

Stream

A stream can be defined as a sequence of data.

There are two kinds of streams:

- **InputStream :** It is used to read data from a source.
- **OutputStream :** It is used to write data to a destination.



➢ Byte Streams

Java byte streams are used to perform input and output of bytes where each byte has 8 bits.

To perform these operations the following classes are used :

- FileInputStream
- FileOutputStream

Example: The contents of data file "Friends.txt" is getting copied into another data file "yourFriends.txt.".

```
import java.io.* ;
class FileCopy
```

```
    {
        public static  void main (String args[]) throws IOException
        {
              FileInputStream Fin = new FileInputStream ("Friends.txt") ;
              FileOutputStream Fout = new FileOutputStream ("yourFriends.txt");
              int c;
              while ((c = Fin.read()) ! = -1)
              {
                    Fout.write (c) ;
              }
              Fin.close ();
              Fout.close ();
        }
    }
```

➢ **Standard Streams**

Java provides three standard streams for data input and output :

- **Standard Input :** This is used to supply data to the user's program and usually a keyboard is used as standard input stream and represented as System.in.
- **Standard Output :** This is used to print the data produced by the user's program and usually a computer monitor is used for standard output stream and represented as System.out.
- **Standard Error :** This is used to output the error data produced by the user's program and usually a computer monitor is used for standard error stream and represented as System.err.

BufferedReader and BufferedWriter

BufferedReader class in Java perfoms reading of text from a character-input stream. Buffered input streams read data from a memory are known as a buffer. The buffer size may be specified. If not, the default size, which is predefined, may be used.

In the same way, BufferedWriter class in Java performs writing of text on a character-output stream. Buffered output streams write data from a memory area known as a buffer.

Operations on Files

Below are the syntax statements for opening a file, using String variable filename :

For creating a new file to write into, open the file using FileWriter.

```
FileWriter F = new FileWriter(filename);
```

For reading from a file, open the file using FileReader.

```
FileReader F = new FileReader(filename);
```

To append data in an existing file, open the existing file using FileWriter with append status true. It appends data at the end of existing data.

```
FileReader F = new FileReader(filename, true);
```

For closing a file

```
FileObject.close();
```

●●

CHAPTER-12

RECURSION

Revision Notes

- A recursive function is a special kind of function that makes a call to itself.
- The function call mechanism in Java supports this possibility, which is known as recursion.
- Every recursive function should have a base case. Upon reaching the base case, the recursion stops and the calculation begins. Without a base case, a recursive function repeats infinite number of times.
- There are two types of recursion as follows:
  (i) **Finite:** Recursive procedure contains a valid test case.
  (ii) **Infinite:** Recursive procedure does not contain any valid test case.
- **Use of Memory Stack in a Recursive Function**
  - ➢ A recursive function uses the memory stack to store its intermediate values (which are calculated after the control returns from the base case). If a recursive function enters infinity (an error situation) then a runtime error occurs, which states "Stack Overflow".

A Function Calling Another Function:

```
main()
{
    ......
    First();
    ......
}
void First()
{
    ......
}
```

Here, main () is the super function and First () is the sub function.

- The control goes to the sub function when it is called and when its code is complete, it goes back to its super function. The super function carries on with the rest of its code after the function call is over.

A Function Calling Itself

- In case of recursive functions, the function becomes its own super and sub function.

e.g.

```
First (argument list a)
{
    ......
    First (argument list b)
    ......
    ......
}
```

Here, First (…) makes a function call to itself.

It is a recursive function.

While making a function call to itself, the value of the arguments identify the super and the sub function.

Features of a Recursive Function

- It makes a call to itself with a change in the arguments.
- It terminates when it reaches its base case.
- It does not use a variable to store the intermediate value instead uses the memory stack of the computer system.
- In case of memory shortage, it gives an error "Stack Overflow".
- Each recursion performs a fresh memory allocation for its arguments/variables.

- Memory requirement of a recursive function is high, hence the system cost also becomes high.
- Certain kind of programs cannot be done without using recursion.

Comparison between Recursion and Iteration

- Recursion using a recursive function and iteration using a loop are meant for doing repetitive tasks.
- Space and time complexity of recursion is generally more than iteration.
- In general, the space requirement for recursion is more compared to iteration.

  e.g.

```
//Iteration
int FactorialLoop (int n)
{
    int  f = 1;
    for(int j = 1; j <= n; j++)
    {
        j = j * j;
    }
}
```

```
//Recursion
int FactorialRecur(int n)
{
    if (n <= 1)
    {
        return 1;
    }
    return(n*FactorialRecur (n – 1));
}
```

- In iteration, first the calculation is done and then the control moves ahead but in recursion, the calculation is done after the base case is reached.
- **Recursive GCD Function**

```
int GCD (int a, int b)
{
    if (a = = b)
    {
          return b;
    }
    else if (a > b)
    {
          GCD (a – b, b);
    }
    else
    {
          GCD (a, b – a);
    }
}
```

Conversion from Decimal to Binary Using Recursion

```
void decToBin(int n)
{
    if (n > 0)
    {
          decToBin (n/2);
          System.out.println("% d", n % 2);
    }
}
```

CHAPTER-13

IMPLEMENTATION OF ALGORITHMS TO SOLVE PROBLEM

Revision Notes

- An algorithm is a stepwise detail of a process. Every process is made up off smaller steps and explaining the process with the help of these smaller steps forms its algorithms.

  e.g. Consider the steps of making tea

  step 1 : Boil water

  step 2 : put fire off and add tea leaves to the water

  step 3 : when the color of water changes and nice aroma comes out, stain the leaves and pour the tea in cups.

  step 4 : serve the tea along with sugar and snacks.
- **Need of an algorithm**
  - Every process needs an explanation in simple language. If the algorithm of a process is not written in a clear and proper manner it will become very difficult to put that task into action. Every beginner needs an algorithm to start a task.
  - Every program has an algorithm. The algorithm written for a program is preferred to be independent of the programming language.
  - It should be written in simple language.
  - Algorithm are also very helpful to do the initial analysis of a problem. Programmers working on big projects first do the problem analysis, then they make the algorithm of the solution and after a satisfactory algorithm is approved, they start working on the problem.
- **Algorithm to find prime number**

```
step 1 : Start
step 2 : Read number n
step 3 : Set f= 0
step 4 : For i = 2 to n – 1
step 5 : If n mod 1 = 0 then
step 6 : Set f = 1 and break
step 7 : Loop
step 8 : If f = 0 then
             print 'The given number is prime'
           else
             print 'The given number is not prime'
step 9 : Stop
```

- **Various Forms Of Algorithm**

  Among the other equivalents of algorithms are pseudocode and Flowchart.
  - **Pseudocode :** It is an algorithm that is very close to a program. It allows many program syntax – like statements. However, a pseudocode is not a program and so, it is not executable.
  - **Flowchart :** It is a diagrammatic or pictorial representation of the steps of doing some work, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.
- **Properties of Algorithm**
  - **Input :** Should be taken from specified set of objects.
  - **Output :** At least one quantity is produced which is related to inputs.
  - **Finiteness :** Algorithm must terminates after a finite number of steps.
  - **Definiteness :** Each instruction must be clear.
- **Complexity of Algorithm**
- There are two types of complexity of algorithm are as :

- ➢ **Space complexity :** Amount of memory needs to run to completion. E.g. space complexity of insertion sort is O(1).
- ➢ **Time Complexity :** Time complexity is three types as
- ➢ **Best case time complexity :** Minimum amount of time computer needs to run to completion. E.g. Best case time complexity of insertion sort is O(n).
- ➢ **Worst Case time complexity :** Maximum amount of time computer needs to run to completion. E.g. Worst case time complexity of insertion sort is $O(n^2)$.
- ➢ **Average case time complexity :** Average amount of time computer needs to run to completion. E.g. Average case time complexity of insertion sort is $O(n^2)$.

o **Asymptotic Notation**

Asymptotic notations are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value. For example: In bubble sort, when the input array is already sorted, the time taken by the algorithm is linear i.e. the best case. To describe the asymptotic running time of an algorithm, we use following notations:

O Notation (upper bound)

Φ Notation (theta bound)

Ω Notation (lower bound)

●●

CHAPTER-14

TRENDS IN COMPUTING AND ETHICAL ISSUES

TOPIC-1
Trends in Computing Artificial Intelligence

Revision Notes

- o **Artificial Intelligence**, commonly known as AI, is the file of computer science which focuses on the creation of intelligent technologies that can think, work and respond like human beings.
- o Intelligence comprises of many non–tangible factors such as learning, reasoning, perception, problem solving etc.
- o Intelligent systems are embedded with artificial intelligence which extend many helpful features.
- o The purpose of the development of AI is to help human beings to have an easier and comfortable life.
- o It focuses on overcoming the common flaws that human beings make in general such as forgetting dates and events, slow calculations and response, confusion and miscalculation etc.
- o The most implying application of artificial intelligence is robotics. There are many other application areas of AI. Some of them are as follows:
  - **(i)** Image and sound identification
  - **(ii)** Temperature and location sensors
  - **(iii)** Decision making
  - **(iv)** Learning and planning
  - **(v)** Problem solving
  - **(vi)** Calculated predictions
- o John McCarthy is known as the Father of Artificial Intelligence.
  - ➢ Development of technology is continuously creating wonders in the field of AI as more and more developed systems are being invented.
  - ➢ Driverless vehicles, home security systems, 360 degree video cameras are to name a few and the list of AI supported systems may seem endless today.
- o **Internet of Things (IoT)**

- It is the network of different devices that can interact with each other and can exchange data and information. These devices can be vehicles, home appliances, electronic gadgets, sensors and other items.
- Each thing is uniquely identifiable through their embedded computing system. These systems are capable of communicating with other systems on their own and hence, require less human intervention.
- Internet of Things provides a way to connect objects to one another through the internet. This network can be controlled remotely which can effectively make modern life more efficient and economical.
- Few of the many application areas are smart homes, automated factories, intelligent transportation and so on.

○ **Advantages of IoT**

○ IoT allows devices to be controlled remotely across the Internet, thus creating opportunities to directly connect and integrate the physical world to the computer based systems using sensors and Internet. There are many advantages of IoT, few are given below:

(i) IoT network benefits are not limited, it has a wide range i.e., individuals, society, stakeholders of businesses etc., due to the fact that IoT network saves time and money.

(ii) It is used for patient health monitoring i.e., various types of wireless sensors are installed on the patient's body which communicate with the IoT network and provides all the required information of the patient under treatment.

(iii) IoT concept is also used in home security devices which are monitored and controlled either locally or remotely using easy to use applications available on mobile phones or smartphones.

Typically IoT devices like security alarm, camera, sensors, door locks etc, are used in automation environment.

(iv) IoT in used is asset and individual tracking, inventory control, energy conservation etc.

○ **Virtual Reality**

- Virtual Reality (VR) means experiencing things through computers that don't exist in the real world.
- It is a computer simulated reality in which a user can interact with replicated real or imaginary environments.
- The experience is totally immersive by the means of visual, auditory and haptic (tough) stimulation so, the constructed reality is almost indistinguishable from the real deal.

○ **Augmented Reality**

- Augmented Reality (AR) is a technology that superimposes a computer generated image on the user's view of the real world, thus providing a merged view.
- Unlike virtual reality, which creates a totally artificial environment, augmented reality uses the existing environment and overlays new information on top of it.

○ **Area in which Virtual/Augmented reality application are commonly used are**

- Design Evaluation (Virtual Prototyping)
- Architectural walk through
- Planning and maintenance
- Concept and data visualizations
- Operations in hazardous environments
- Training and simulation
- Sales and marketing
- Entertainment and leisure

○ **Ethical Issues in the Field of Computing**

- Ethical issues in the field of computing provide a guideline to proper and correct usage of computers and its applications. Many social responsibilities should be shouldered by the citizens to ensure that the people who invent and develop useful software and other applications are not deprived of their basic rights.

○ **Intellectual Property**

- Intellectual property is the result of creativity of the human intellect which is intangible in nature.
  Some examples are musical notes, literary work or artistic creation and so on.
- Some theories and methods can also be included in intellectual property. Computer software is considered as an intellectual property because of its intangible nature.

○ **Intellectual Property Laws and Rights**

- There are certain Laws and Rights that guard Intellectual properties. Listed below are some of the major Intellectual property rights (IPR).

(i) Copyright: A copyright is a legal right permitting a creator exclusive rights for the use of this creation. It gives the right to the creator so that no one can copy his work. However, someone can do the whole thing from the beginning keeping the basic idea same. Copyrighted items have a symbol of ©.

(ii) Patent: It protects an idea. No one can work on a patented idea even if the whole plan is new.

(iii) Trademark: It is distinct name that is used for some kind of goods or services. Symbol of trademark is ®.

(iv) Trade secrets: These are certain formulas or plans that are kept confidential for keeping the uniqueness of an item/property.

- **How are Intellectual Property Rights Helpful?**

  Intellectual Property Rights provide legal protection and security to the creators of Intellectual Property.

  Intellectual properties are of immense help to the society for its overall improvement.

- **Software Copyright and Software Patent**
  - A software copyright is a legal right permitting a creator of literary and artistic works exclusive rights for its use, while a software patent is an exclusive right granted to an inventor for new and useful inventions.
  - A software copyrighted work is protected right after its creation, while an invention will be protected only after the software patent is issued.
  - Parts of a software copyrighted work can be used by another person, provided it is acknowledged, while a patented work cannot be made, used or sold by another, without permission from the software patent holder.
  - A software copyright costs less than software patent and are easier to acquire.
  - A software copyright lasts generally for 60 years, while a software patent will expire 10-20 years after it is issued.
  - Both software copyright and software patent owner can take legal action against any person who infringes to copyright/patent.

- **Software Licensing**
  - Software licensing is the permission given to an individual or a group to use the software. Before using any software, the user must ensure that the software is a licensed software.

- **Software license are of various types, some are listed below:**
  - A software license may allow a software to be used in only one machine.
  - A software license may allow a software to be used in a specific number of machines.
  - A software license may allow a software to be used in a given number of machines but at the same time only a single machine can work on it.

- **Software Piracy**
  - Piracy is the illegal copying of a software. By buying any software, a buyer does not become the owner of that software. He only becomes a licensed user of that software and does not get the right to sell/give his copy to others. Each and every user must buy the software from its owner and not from its user.

- **Harms Caused by Software Piracy**

  Software piracy is unethical. Some of the harms caused are:
  - Financial loss to the creative programmers, who make them.
  - Causes hike in the general cost of the software.
  - Programmers lose interest in making new creative software fearing the loss caused by piracy.
  - Software companies loose their profits as people purchase pirated software in place of the original software.

- **Free Software Foundation**
  - The Free Software Foundation (FSF) is a non- profit organisation that supports the free software movement. It was founded by Richard Stallman on 4th October, 1985. The Free software Movement promotes the freedom to create, distribute and modify software with the creator's permission. It is non- commercial in nature.

- **Open Source Software**
  - Open Source Software (OSS) is a computer software whose source code is open to all. Any person can read, debug, modify and can sometimes even distribute the open source software.

- **Various types of Licensing (e.g. GPL, BSD)**

  A software license is a legal document that manages the proper usage of the software by its creator and its users.
  - **General Public License (GPL)** is a free license that allows the software to be sold/used as a commercial object.
  - **Berkeley Source Distribution (BSD)** is another class of license that provides some protection to commercial use of a software, such as giving due credit to the creator of a software, protect the creator from certain non functioning part of a software etc.

Know the Terms

- **DRM :** Digital Rights Management
- **WTO :** World Trade Organisation
- **IPR :** Intellectual Property Rights
- **OSS :** Open Source Software
- **FSF :** Free Software Foundation

TOPIC-2
Cyber Security

Revision Notes

- **Netiquette**

  Etiquette to be followed while using the Internet, is called Netiquette.

  With the Internet proving to be so very useful to us, we should also be careful and sincere from our side.

  Ignorant or careless attitude may prove to be harmful to the society and can cause unnecessary trouble to millions of users who use the Internet.

- **What we should not do over the Internet**
  - We should not use the computer to harm other people in any way.
  - We should not interfere with other people's work.
  - We should not disturb someone who is working on something important on the computer.
  - We should not spread wrong or harmful messages through the Internet.
  - We should not go to those websites for which we do not have permission.
- **What we should do over the Internet**
  - We should be helpful to people who are learning.
  - We should always use proper language and correct information in our messages.
  - We should always write messages with proper subject so that if the receiver is busy, he can decide whether he should give immediate attention to our message or can he read it later.
  - We should be proud of our own culture and should respect the cultures from all over the world as well. It will help in developing a beautiful relationship with people from all over the world through the Internet.
  - We should stop our friends from doing wrong things on the computer.
- **Email Etiquettes**

  Etiquette means good manners. As a responsible citizen, one should maintain certain email etiquettes.
- **A person should not do the following over the Email**
  - Ask or acquire other's email password.
  - Open/use other's email account.
  - Forward gossips/false news via email.
  - Forward meaningless mails to others.
  - Send mails without subject as subject helps the receiver to know its importance.
- **A person should do the following over the Email**
  - Help new users.
  - Use bcc in case a mail has to be sent/forwarded to various people.
  - Remove the chain of addresses in case of multiple forwarding.
  - Send/reply formal mails by maintaining the standards.
  - Remember that alphabets in upper case indicate loudly spoken words. Avoid uppercase if not necessary.
- **Various Security Threats**

  Computers these days are subject to various security threats, which can cause many different types of damages. Some security threats are given below:

- **Virus**
  Malicious computer programs that are often sent as an email attachment or a download with the intent of infecting your computer, as well as the computers of everyone in your contact list. Just visiting the site can start an automatic download of the virus.
- **What they can do:**
  - Send spam
  - Provide criminals with access to your computer and contact lists
  - Scan and find personal information like passwords on your computer
  - Hijack your web browser
  - Disable your security settings
  - Display unwanted ads.
- When a program is running, the virus attached to it could infiltrate your hard drive and also spread its USB keys and external hard drives.
- **Hacking**
  It is a term used to describe actions taken by someone to gain unauthorized access to a computer. The availability of information online about the tools, techniques, and malware makes it easier for even non-technical people to undertake malicious activities.
- **What it can do:**
  - Find weaknesses in your security settings and exploit them in order to access your information.
  - Install a Trajan horse, providing a back door for hackers to enter and search for your information.
- **Malware**
  It is one of the most common ways to infiltrate or damage your computer.
  Malware software that infects your computer, such as computer viruses, worms, Trojan horses, spyware and adware.
- **What it can do:**
  - Intimidate you with scare ware, which is usually a pop – up message that tells you, your computer has a security problem or other false information.
  - Reformat the hard drive of your computer causing you to lose all your information.
  - Alter or delete files
  - Steal sensitive information
  - Send emails on your behalf
  - Take control of your computer and all the software running on it.
- **Phishing**
  Phishing is used most often by cyber criminals because it is easy to execute and can produce the results they are looking for with very little effort.
  Fake emails, text messages and websites created to look like they are from authentic companies. They are sent by criminals to steal personal and financial information from you. This is also known as spoofing.
- **What it can do:**
  - Trick you into giving them information by asking you to update, validate or confirm your account.
  - Provides cyber criminals with your username and passwords so that they can access your accounts and steal your credit card numbers.
- **Spam**
  Spam is one of the most common methods of both sending information out and collecting it from unsuspecting people.
  The mass distribution of unsolicited messages, advertising or pornography to addresses which can be easily found on the Internet through things like social networking sites, company websites and personal blogs.
- **What it can do:**
  - Annoy you with unwanted junk mail.
  - Create a burden for communications service providers and businesses to filter electronic messages.
  - Phish for your information by tricking you into following links or entering details with too good to be true offers and promotions.
  - Provide a vehicle for malware, scans, fraud and threats to your privacy.

- **Trojan Horses**

  Trojan horse is a malicious program that is disguised as, or embedded within, legitimate software. It is an executable file that will install itself and run automatically once it's downloaded.
- **What it can do:**
  - Delete your files
  - Use your computer to hack other computers
  - Watch you through your web cam
  - Log your keystrokes
  - Record usernames, password's and other personal information
- **Worms:**

  Worms are common threat to computers and the Internet as a whole. A worm, unlike a virus, works on its own without attaching itself to files or programs.

  It lives in your computer memory, does not damage or alter the hard drive and propagates by sending itself to the computers in a network.
- **What they can do:**
  - Spreads itself to everyone in your contact list.
  - Cause a tremendous amount of damage by shutting down parts of the Internet, wreaking havoc on an internet network and costing companies enormous amounts of lost revenue.
- **Privacy**

  It is the right to be free from secret surveillance and to determine whether, when how and to whom, one's personal or organisational information is to be revealed. In specific, privacy may be divided into four categories as follows:

  (i) Physical: restriction on others to experience a person or situation through one or more of the human senses.

  (ii) Informational: restriction on searching for or revealing facts that are unknown or unknowable to others.

  (iii) Decisional: restriction on interfering in decisions that are exclusive to an entity.

  (iv) Dispositional: restriction on attempts to know an individual's state of mind.

  - A privacy policy, in the context of IT, is a document that tells readers how a technology or the product or service will use their personal information. The term privacy policy is now often used in IT because so many IT products and systems gather and use personal information from users in so many different ways.
  - A privacy policy can be printed on paper, displayed on a monitor or device screen or hosted on a website. It will detail the ways in which a company will use various kinds of information about users, which generally includes some combination of demographic data more personal details. A privacy policy should cover the range of information the user gives to the technology, whether that includes identifiers such as name, social security number or other I.D., financial information, medical information etc.
  - Privacy policies have become important parts of E-commerce and other services agreements. They will continue to play a vital role in digital and online business wherever corporations or other parties gather sensitive information.

●●