PROGRAM

A program is a set of statements or instructions used to solve a problem.

PROGRAMMING

→ The art of writing a program is called as programming.

→ Art refers to the efficient writing of program.

→ Line of code & Efficient program.

PROGRAMMING LANGUAGES

The languages which are used for programming is called programming languages.

Programming languages are divided into two types

(1) Low level language
(2) High level language.

Low level language

A language that can easily understand by the machine is called low level language.

Low level language is further classified into two types.

(1) Machine language
(1) Assembly language.

Machine Language

It is also called 1st generation programming language.

In this language instructions or statements written in the form of 1's and 0's.

Ex: 1010111000, 1001001001

Advantages:

(1) Easily understandable by machine.
(2) No translator is required.

Disadvantages

(1) Difficult to understand by human.
(2) Difficult to write.
(3) Difficult to identify the error & modify.
(4) Machine dependent (lack of portability).

Assembly Language

13.10.23

It is also called 2nd generation language.

In this language some Mnemonics are introduced, i.e. ADD, SUB, MUL, DIV, MOV etc.

(Q) Write a program to add two number in Assembly language.

```
MOV A,4  → store 4 in 'A'
MOV B,5  → store 5 in 'B'
ADD A,B  → Add 'A' & 'B' and store in 'A'
MOV sum,A → store value of 'A' in 'sum'
```

Advantages:

→ Easily understable by human.
→ Easy to write, modify, debug.

Disadvantages:

→ Mnemonics are not standardized.
→ Machine dependent
→ Need a translator.

High Level language

→ High level languages are like english-like statements.
→ We can write by using alphabets, digits, punctuations, special characters like @, #, $, 1, ! etc.

(Q) Write a program to find sum of two numbers.

```
Read a,b
Sum = a+b
Print sum
```

Advantages

→ Easy to understand, remember, write, modify and debug the errors.
→ Machine independent.

Disadvantages

→ Need of translator.

→ High level languages are classified into
(1) General Purpose High Level Language
(2) Specific Purpose High Level Language.

General purpose H.L.L.

Ex: C, Pascal, BASIC (Beginner's All Purpose Symbolic Instruction code)

Specific purpose H.L.L.

Ex: COBOL (Common Business oriented Language) → Business application
FORTRA (Formula Translation) → Scientific Application.
C++/JAVA/.NET → Desktop based application, web based application, Mobile Application.

Python/R/PROLOG (Programming in Logic) → AI based application

SOFTWARE

→ A software is a program or collection of programs.
→ The software can't be touched or sensed.
→ The software will make the hardware to run.
→ Softwares are classified into 3 major categories
(1) Application software
(2) System software
(3) Utility software.

Application Software

→ It is a collection of programs written by an user (user means who knows the program) to solve a problem in his particular area of interest.

→ Ex: Library Management System, Student Information System, Payroll System, Inventory system etc. are the examples of application software.

System Software

→ It is a collection of programs meant for computer system Management.

→ These programs are written by the manufacturer.

→ System softwares are supervisory programs for the application software that are run in our system because system softwares are responsible for the application software to be run & work smoothly and efficiently.

→ System softwares are classified into

(i) Operating System
(ii) Language processors
(iii) System utilities.

Operating System

→ It is an integrated collection of programs which makes the hardware to work or operate and allows the user to run his applications smoothly.

→ It also acts as an interface between the man and machine.

→ Operating system manage different resources i.e. processor, memory devices (RAM, ROM), I/O devices (keyboard, Mouse, Monitor), files etc

→ First operating system that was developed -> DOS (Disk Operating System) DOS was developed by IBM and later acquired by Microsoft and renamed as "MS-DOS".

→ Some other operating systems are Unix-OS, Linux (RedHat, Mondrek, Debian, SUSE, Ubuntu, Bharat), Windows (98, xp, 7, 8, 10, 11), Windows server 20[illegible] 2023, Windows NT, Macintosh (Mac OS).

Language Processors

14.10.23

→ It is a program or translator which converts high level programming language to machine level programming language.

→ This language processor comes in 3 different categories

(i) compiler
(ii) Interpreter
(iii) Assembler.

Compiler

→ It is a program or translator which reads all the input statements of an high level language at a time and converts them into a machine level language.

→ During this process it displays the errors if there are any and are rectified by the programmer.

→ Once the errors are corrected, then the source program has to be recompiled.

→ The process of converting a high level program into a machine language using a compiler is called compilation.

→ Ex: Turbo c/c++, Borland c++ compiler, Java compiler and GNU compiler.

Interpreter

→ It is a program or translator which converts a high level language program into a machine language statement by statement.

→ During this process if there is any error at a particular statement, the programmer needs to rectify it, otherwise the interpreter will not proceed to the next statement.

→ The process of converting high level language into machine language using interpreter is called interpretation.

→ Ex: BASIC, BASICA, Java Interpreter.

Assembler

→ It is a program which translates assembly level language into machine language.

→ The process of converting assembly language into machine language using an assembler is known as assembling.

→ Ex: TASM (Turbo Assembler), MASM (Macro Assembler).

System Utilities

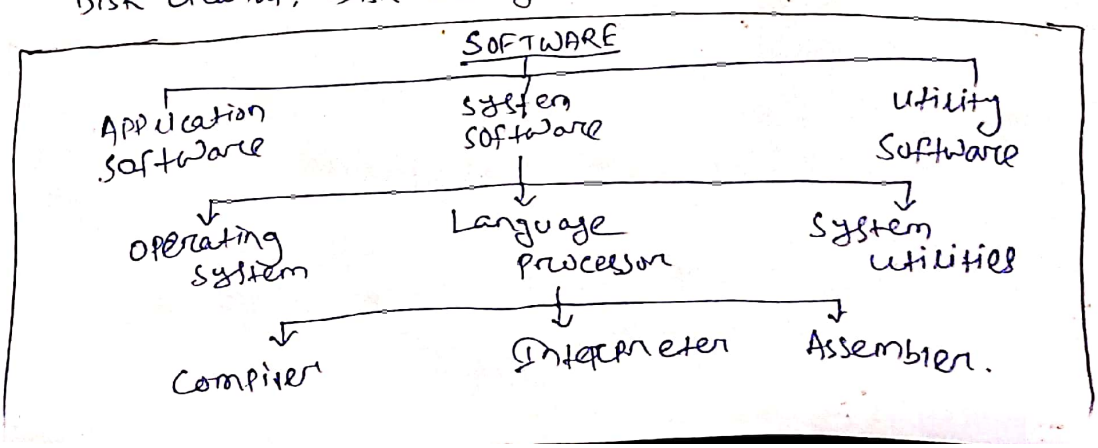
→ These are the programs used in linking and loading of an object program. (a program which is in machine readable form is called object program).

→ Ex: Loaders and Linkers.

Utility Software

→ Application software that assist Operating System in carrying out certain specialized tasks are called utility software.

→ Ex: Antivirus, File management tools like Windows Explorer, Google desktop, File compression tools like WinRar, WinZip, Disk cleanup, Disk Defragmentation, Backup etc.



SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

There are some steps involved in problem solving. The steps are:

(1) Problem Definition / Requirement Gathering
(2) Problem Analysis
(3) Designing / problem solution approaches.
(4) Coding
(5) Debugging / Testing
(6) Deployment
(7) Maintenance

Problem Definition

This is the first step in computer problem solving

The problem solver should understand the problem thoroughly in terms of the requirements (Input and Output).

The problem solver should extract from the problem statement, a set of well-defined and precise tasks that can be carried out.

If the problem is not properly interpreted then we cannot obtain the desire results.

Ex: If a student appearing for an exam and he doesn't understand the question, is he able to answer the question? Definitely: NO.

Problem Analysis

The given problem must be analysed before it is solved.

This determines the data items, their types and relationship.

The types of operations (arithmetic and logical), amount of memory required and time needed for execution should be clearly specified.

Ex: Once the student understands the question, he needs to analyse the scope of the question and the topics to be written under it.

Designing / problem solution approaches

It is essential to devise a solution for different kinds of problems.

Algorithm, Flowchart and Pseudocode are the solution techniques used for solving the problem.

Algorithm

An algorithm is a solution which is represented in natural language.

An algorithm is a step-by-step procedure written in english-like statement for solving a problem.

Algorithms are of 3 types

(1) Sequential Algorithms
(2) Conditional Algorithms
(3) Repetitive Algorithms.

Algorithms are never executed but must have an logic to approach the solution.

Sequential Algorithm

→ If the statements are executed one after the another in a sequence then we called them as a sequential algorithm.

→ Ex-1: Write an algorithm to perform basic arithmatic operation on 2 variables. a & b.

→ Step-1: Start
Step-2: Read the values of a and b.
Step-3: Addition of two numbers is sum = a+b
Subtraction of two numbers is sub = a-b
Multiplication of two numbers is mul = a*b
Division of two numbers is Div = a/b
Step-4: print sum
print sub
print mul
print Div
Step-5: Stop

16.10.23

Ex-2: Write an algorithm to find the sum and average of 3 numbers.

→ Step-1: Start
Step-2: Read the values of a, b and c.
Step-3: Sum = a+b+c
Step-4: Average = $\left(\frac{a+b+c}{3} \text{ or } \right) \frac{sum}{3}$
Step-5: Print sum and Average
Step-6: Stop.

Ex-3: Write an algorithm to find the area and circumference of a circle.

→ Step-1: Start
Step-2: Read the value of radius say r.
Step-3: Area = πr^2 = 3.142 x r x r
Step-4: circumference = $2\pi r$ = 2 x 3.142 x r
Step-5: Print area and circumference
Step-6: Stop.

Ex-4: Write an algorithm to find the simple interest by reading principle amount, time period and rate of interest.

→ Lets say principle amount = P Time period = T
Rate of interest = R

Step-1: Start
Step-2: Read the values of P, T and R
Step-3: Simple Interest SI = $P \times T \times R / 100$
Step-4: print SI
Step-5: Stop.

Ex-5: Write an algorithm which reads a temperature in Fahrenheit and converts into celsius.

→ Let say temperature in fahrenheit = FT
temperature in celsius = CT

$$CT = (FT - 32) \times \frac{5}{9} \qquad FT = (CT \times \frac{9}{5}) + 32$$

Step-1: Start
Step-2: Read the value of FT.
Step-3: $CT = (FT - 32) \times \frac{5}{9}$
Step-4: Print CT
Step-5: Stop.

Ex-6: Write an algorithm which reads two numbers and swap these two numbers using a temporary variable.

→ Step-1: Start
Step-2: Read the values of a and b.
Step-3: Take a temporary variable as 'c' and assign the value of 'a' to it i.e. c=a.
Step-4: a = b
Step-5: b = c
Step-6: Print a and b.
Step-7: Stop

Ex-7: Write an algorithm which reads two numbers and swap these two numbers without using a temporary variable.

→ We can solve this problem by using arithmatic operation and by using bitwise XOR operator also.

Using arithmatic operation

Step-1: Start
Step-2: Read the values of a and b
Step-3: a = a+b
Step-4: b = a-b
Step-5: a = a-b
Step-6: Print a & b
Step-7: Stop.

Using Bitwise XOR

Step-1: Start
Step-2: Read the values of a and b
Step-3: a = a ^ b
Step-4: b = a ^ b
Step-5: a = a ^ b
Step-6: Print a and b
Step-7: Stop.

Conditional Algorithms

→ In conditional algorithms we make certain decisions based on the truthness of the condition (true/false)

Ex-1: Write an algorithm which reads the values of a and b and check whether they are equal or not.

→ Step-1: Start
Step-2: Read the values of a & b.
Step-3: If a = b, print "a and b are equal"
otherwise print "a and b are not equal."
Step-4: Stop.

Ex-2: Write an algorithm to check whether the given number is even or odd.

→ Step-1: Start
Step-2: Read the number say n.
Step-3: Divide n by 2 and store the remainder in rem.
Step-4: If rem = 0, print "the number is even".
otherwise print "the number is odd".
Step-5: Stop

Ex-3: Write an algorithm to check whether the given number is positive or negative number.

→ Step-1: Start
Step-2: Read the number say n
Step-3: If n>0, print "the number is positive number".
otherwise if n<0, print "the number is negative number".
Step-4: Stop.

Ex-4: Write an algorithm which reads the age of a person and check whether he is eligible for vote or not.

→ Step-1: Start
Step-2: Read the age of the person say 'a'.
Step-3: If a >= 18, print "the person is eligible for vote."
Otherwise, print "the person is not eligible for vote."
Step-4: Stop.

Ex-5: Write an algorithm which reads a & b and check whether a is greater than b or not.

→ Step-1: Start
Step-2: Read the values of a & b.
Step-3: If a>b, print "a is greater than b".
otherwise if a<b, print "a is not greater than b".
Otherwise print "a is equal to b".
Step-4: Stop.

Ex-6: Write an algorithm to find the largest among the three numbers.

→ Step-1: Start
Step-2: Read the values of a, b & c
Step-3: If a>b then go to Step-5
Step-4: If b>c then Print "b is largest"
Otherwise Print "c is largest."
Step-5: If a>c then print "a is largest"
otherwise print "c is largest"
Step-6: Stop.

Ex-7: Write an algorithm to print the grade of a student based upon the marks secured in the exam.

Marks	Grade
>= 75	A
>=65 & <75	B
>=55 & <65	C
>=45 & <55	D
>=35 & <45	E
<35	F

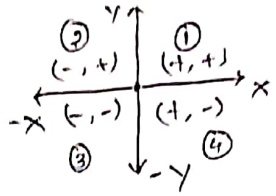
→ Step-1: Start
Step-2: Read the mark say m.
Step-3: If m>=75, print 'grade=A'
otherwise if m>=65 & m<75, print 'grade=B'
Otherwise if m>=55 & m<65, print 'grade=C'
otherwise if m>=45 & m<55, print 'grade=D'
Otherwise if m>=35 & m<45, Print 'grade=E'
Otherwise print 'grade=F'.
Step-4: Stop.

Ex-8: Write an algorithm which reads the height of a person and determine whether he is dwarf, average height, taller or abnormal height.

dwarf < 165cm
average height >= 165 cm and < 175cm
taller >= 175cm & < 180 cm
abnormal height >= 180cm.

→ Step-1: Start
Step-2: Read the height of a person in cm say h.
Step-3: If h<165, Print "Dwarf"
Otherwise if h>=165 & h<175. Print "Average Height"
Otherwise if h>=175 & h<180, Print "Taller".
Otherwise, Print "Abnormal height.
Step-4: Stop.

Ex-9: Write an algorithm which reads the value of X & Y as a coordinate system and determine the quadrant.



→ Step-1: Start
Step-2: Read the value of X and Y.
Step-3: If X>0, then go to step-4
Otherwise if X<0, then go to step-5
Otherwise if X=0, then go to step-6
Step-4: If Y>0, then print "First Quadrant"
Otherwise if Y<0, then print "Fourth Quadrant"
Otherwise if Y=0, then print "positive side on X-axis".
Step-5: If Y>0, then print "Second Quadrant"
Otherwise if Y<0, then print "Third Quadrant"
Otherwise if Y=0, then print "Negative side on X-axis".
Step-6: If Y>0, then print "positive side on Y-axis".
Otherwise if Y<0, then print "Negative side on Y-axis".
Otherwise if Y=0, then print "Origin".
Step-7: Stop.

17.10.23

Repitive Algorithm

In repitive algorithm, certain steps are repeated for required number of times until the condition is satisfied such that it yields to a solution.

Ex-1: Write an algorithm to find the sum of first N natural numbers.

→ Step-1: Start
Step-2: Read value of N (Let say 5)
Step-3: Set sum=0, i=1
Step-4: sum = sum+i
Step-5: i = i+1
Step-6: If i<=N, then go to step-4
Step-7: print sum
Step-8: Stop.

2<=5	sum = 0+1=1,	i = 1+1 = 2.
3<=5	sum = 1+2=3	i = 2+1=3
4<=5	sum = 3+3=6	i = 3+1=4
	sum = 6+4=10	i = 4+1=5
5<=5	sum = 10+5=15	i = 5+1=6
6<=5 (X)		
	sum = 15	

Ex-2: Write an algorithm to print even numbers between 2 to 50.

-> Step-1: Start
Step-2: Read the value of N (Let say 50)
Step-3: Set i = 2
Step-4: Divide i by 2 and store remainder in rem.
Step-5: If rem = 0, then print i and go to Step-7
Step-6: If rem ≠ 0, then i = i+1 and go to step-4.
Step-7: i = i+2
Step-8: print i
Step-9: If i <= N-2, go to step-8
Step-10: Stop.

(or)

Step-1: Start
Step-2: Read the value of N (say 50)
Step-3: Set i = 2
Step-4: Divide i by 2 and store remainder in rem
Step-5: If rem ≠ 0, then go to step-7
Step-6: print i
Step-7: i = i+1
Step-8: If i <= N, then go to step-4
Step-9: Stop.

Ex.-3: Write an algorithm to find the factorial of a given number.

-> Step-1: Start
Step-2: Read the value of N (Let say 5)
Step-3: Set Mul = 1, i = 1
Step-4: Mul = Mul x i
Step-5: i = i+1
Step-6: If i <= N, go to step-4
Step-7: Print Mul
Step-8: Stop.

mul = 1, i = 1, mul = 1x1 = 1, i = 1+1 = 2
2<=5 mul = 1x2 = 2, i = 2+1 = 3
3<=5 mul = 2x3 = 6, i = 3+1 = 4
4<=5 mul = 6x4 = 24, i = 4+1 = 5
5<=5 mul = 24x5 = 120, i = 5+1 = 6
6<=5 (x)
Mul = 120.

(or)

Step-1: Start
Step-2: Read the value of N (say 5)
Step-3: Set fact = 1
Step-4: fact = fact x N
Step-5: N = N-1
Step-6: If N > 0 go to step-4
Step-7: Print fact
Step-8: Stop.

fact = 1, N = 5, fact = 5x1 = 5, N = 5-1 = 4
4>0, fact = 5x4 = 20, N = 4-1 = 3
3>0, fact = 20x3 = 60, N = 3-1 = 2
2>0, fact = 60x2 = 120, N = 2-1 = 1
1>0, fact = 120x1 = 120, N = 1-1 = 0
0>0 (x) fact = 120

Ex-4: Write an algorithm to find sum of individual digits of a given number.

→ Step-1: Start
Step-2: Read the value of N (Let say 123)
Step-3: ~~Divide~~ Set sum = 0
Step-4: Divide N by 10 and store the remainder in rem. and quotient in N
Step-5: sum = sum + rem
Step-6: If N > 0, go to step-4
Step-7: Print sum
Step-8: Stop.

sum=0, rem=3, N=12
sum = 0+3 = 3,
12>0, rem=2, N=1, sum=3+2=5
1>0, rem=1, N=0, sum=5+1=6
0>0 (x) sum=6

Ex-5: Write an algorithm to convert a given binary number into a decimal number.

→ Step-1: Start
Step-2: Read the binary number say bnum (Let take 1111)
Step-3: Set dnum = 0, base = 1
Step-4: Divide bnum by 10 and store the remainder in rem.
Step-5: dnum = dnum + (rem x base)
Step-6: Divide bnum by 10 and store the quotient in bnum
Step-7: base = base x 2
Step-8: If bnum > 0, then go to step-4
Step-9: Print dnum
Step-10: Stop.

dnum=0, base=1, rem=1, dnum=0+(1x1)=1, bnum=111, base=2x1=2
111>0, rem=1, dnum=1+(1x2)=3, bnum=11, base=2x2=4
11>0, rem=1, dnum=3+(1x4)=7, bnum=1, base=4x2=8
1>0, rem=1, dnum=7+(1x8)=15, bnum=0, base=8x2=16
0>0(x) dnum=15 Ans

18.10.23

Ex-6: Write an algorithm to convert given decimal number into a binary number.

→ Step-1: Start
Step-2: Read the decimal number say dnum (Let take 15)
Step-3: Set bnum = 0, base = 1
Step-4: Divide dnum by 2 and store the remainder in rem.
Step-5: bnum = bnum + (rem x base)
Step-6: Divide dnum by 2 and store the quotient in dnum.
Step-7: base = base x 10

Step-8: If dnum > 0, then go to step-4
Step-9: print bnum
Step-10: stop.

dnum = 15 bnum = 0 base = 1
rem = 1, bnum = 0 + (1×1) = 1, base = 1×10 = 10, dnum = 7
7 > 0 rem = 1, bnum = 1 + (1×10) = 11, base = 10×10 = 100, dnum = 3
3 > 0 rem = 1, bnum = 11 + (1×100) = 111, base = 100×10 = 1000, dnum = 1
1 > 0 rem = 1, bnum = 111 + (1×1000) = 1111, base = 1000×10 = 10000, dnum = 0
0 > 0 (X) bnum = 1111 (Ans)

Ex-7: Write an algorithm to find the reverse of a given number.

-> Step-1: Start
Step-2: Read the value of N (say 123)
Step-3: Set rev = 0
Step-4: Divide N by 10 and store the remainder in rem.
Step-5: rev = rev × 10 + rem
Step-6: Divide N by 10 and store the quotient in N
Step-7: If N > 0, then go to step-4
Step-8: print rev
Step-9: Stop.

Ex-8: Write an algorithm to check whether the given number is palindrome or not.

-> Step-1: Start
Step-2: Read the value of N (say 414)
Step-3: Set rev = 0, x = N
Step-4: Divide N by 10 and store the remainder in rem
Step-5: rev = rev × 10 + rem
Step-6: Divide N by 10 and store the quotient in N
Step-7: If N > 0, then go to step-4
Step-8: If x = N, then print "x is palindrome"
otherwise print "x is not palindrome."
Step-9: Stop.

Ex-9: Write an algorithm to generate a fibonacci series upto a given limit.

-> Step-1: Start
Step-2: Read the value of limit say N (Let take N = 5)
Step-3: Set fib1 = 0, fib2 = 1, count = 2
Step-4: print fib1, fib2
Step-5: fib3 = fib1 + fib2
Step-6: print fib3.

[P.T.O.]

Step-7: fib1 = fib2, fib2 = fib3
Step-8: count = count + 1
Step-9: If count < N, then go to step-5
Step-10: Stop.

fib1 = 0, fib2 = 1 fib3 = 0+1 = 1 count = 2
fib1 = 1, fib2 = 1, count = 2+1 = 3,
3<5, fib3 = 1+1 = 2, fib1 = 1, fib2 = 2, count = 3+1 = 4
4<5 fib3 = 1+2 = 3, fib1 = 2, fib2 = 3, count = 4+1 = 5
5<5 (X)

Series: 0, 1, 1, 2, 3 Ans

Ex-10: Write an algorithm to check whether a 3-digit number is an armstrong number or not.

Let take a 3 digit no. = 153
$1^3+5^3+3^3 = 1+125+27 = 153$
So 153 is an armstrong number.

→ Step-1: Start
Step-2: Read the 3-digit number say N
Step-3: Set sum = 0, temp = N
Step-4: Divide N by 10 and store the remainder in rem.
Step-5: Sum = Sum + (rem x rem x rem)
Step-6: Divide N by 10 and store the quotient in N
Step-7: If N > 0, then go to step-4
Step-8: If temp = Sum, then print "temp is an armstrong number: otherwise print, "temp is not an armstrong number"
Step-9: Stop.

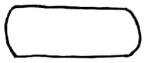
N = 153, sum = 0, temp = 153,
rem = 3, sum = 0+(3x3x3) = 27, N = 15
15>0, rem = 5, sum = 27+(5x5x5) = 152, N = 1
1>0, rem = 1, sum = 152+(1x1x1) = 153, N = 0
0>0 (X)

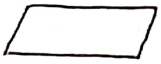
153 is an armstrong number.

Flowchart

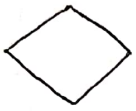
→ A flowchart is a diagrametic representation of an algorithm.
→ It is also called as visual on graphical representation of an algorithm.
→ It is also treated as blueprint to an algorithm.

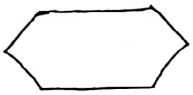
Symbols used in Flowchart

(1)  → Start / Stop

(2)  → Input / Output

(3)  → Processing

(4)  → Decision Making

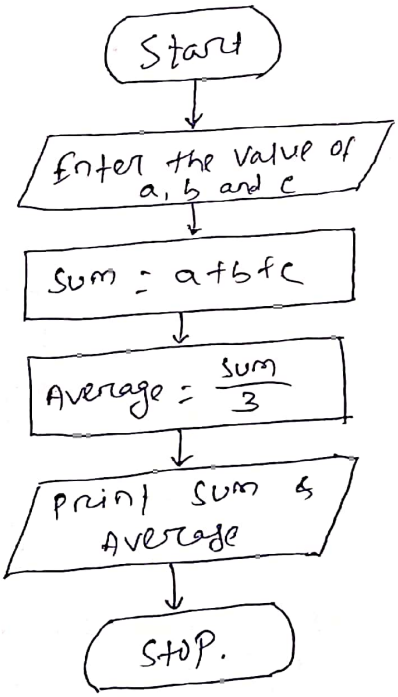
(5)  → Repetition of statement.

(6)  → Connectivity to symbols

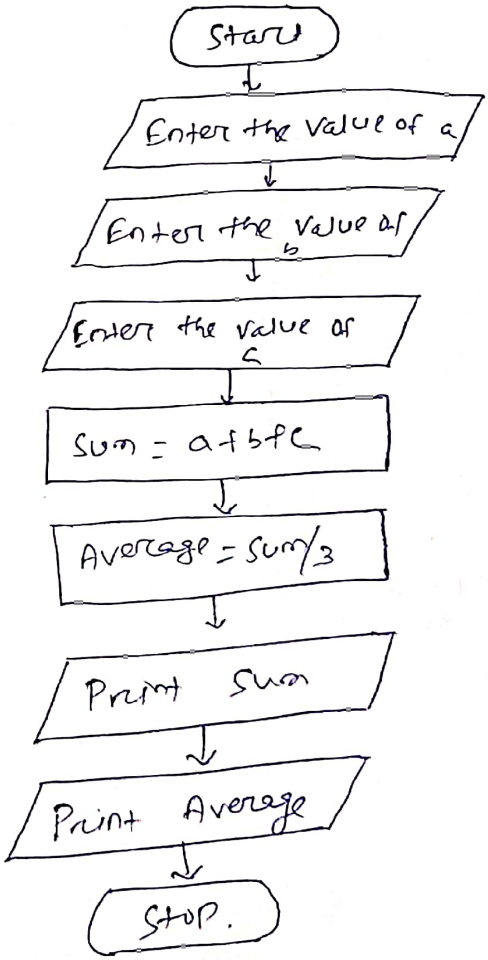
(7)  → Continuation.

19.10.23

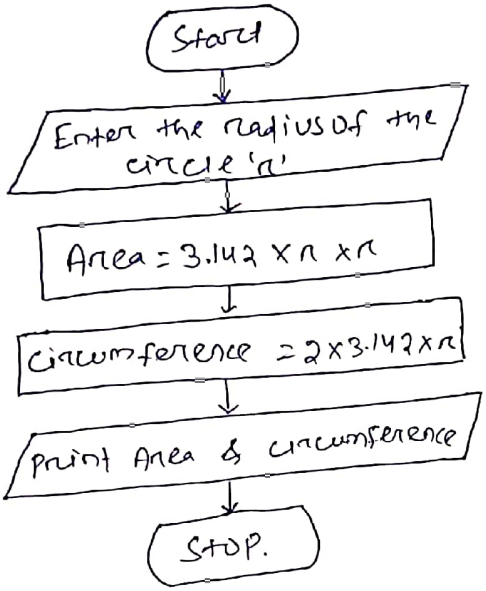
EX-1: Draw a flowchart which reads the 3 numbers and find out their sum and average.



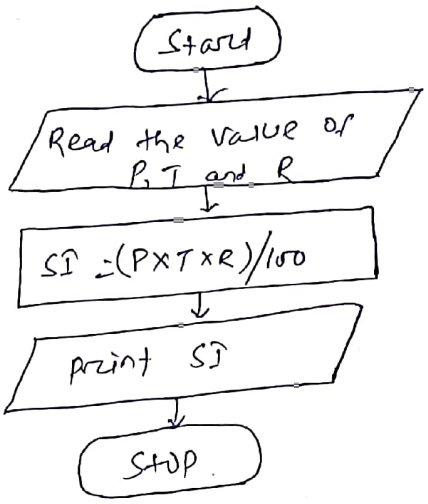
Using Raptor



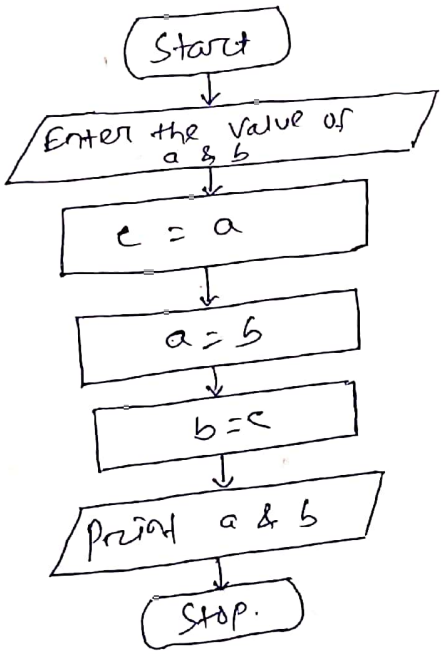
Ex-2: Draw a flowchart to find out the area & circumference of a circle.



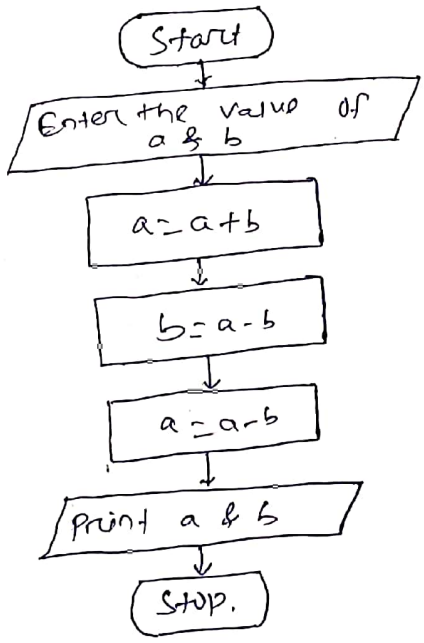
Ex-3: Draw a flowchart to find the simple interest by reading principle amount, time period and rate of interest.



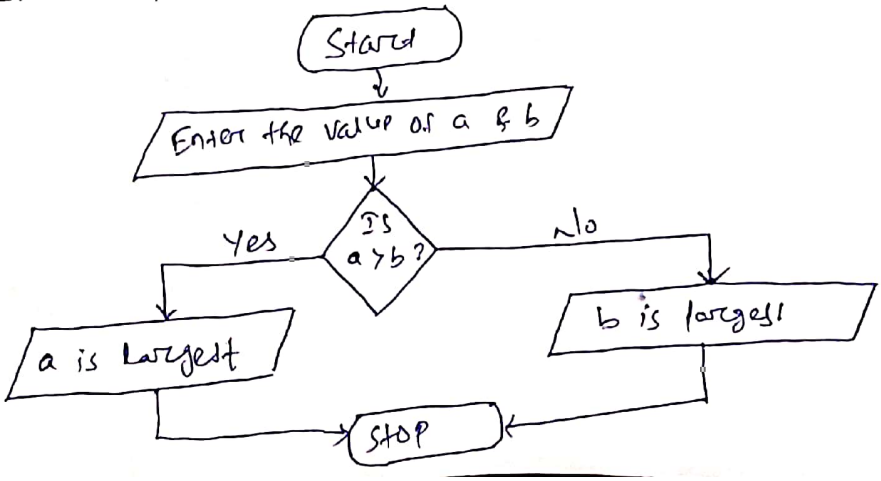
Ex-4: Draw a flowchart to swap the values of two variables using a temporary variable.



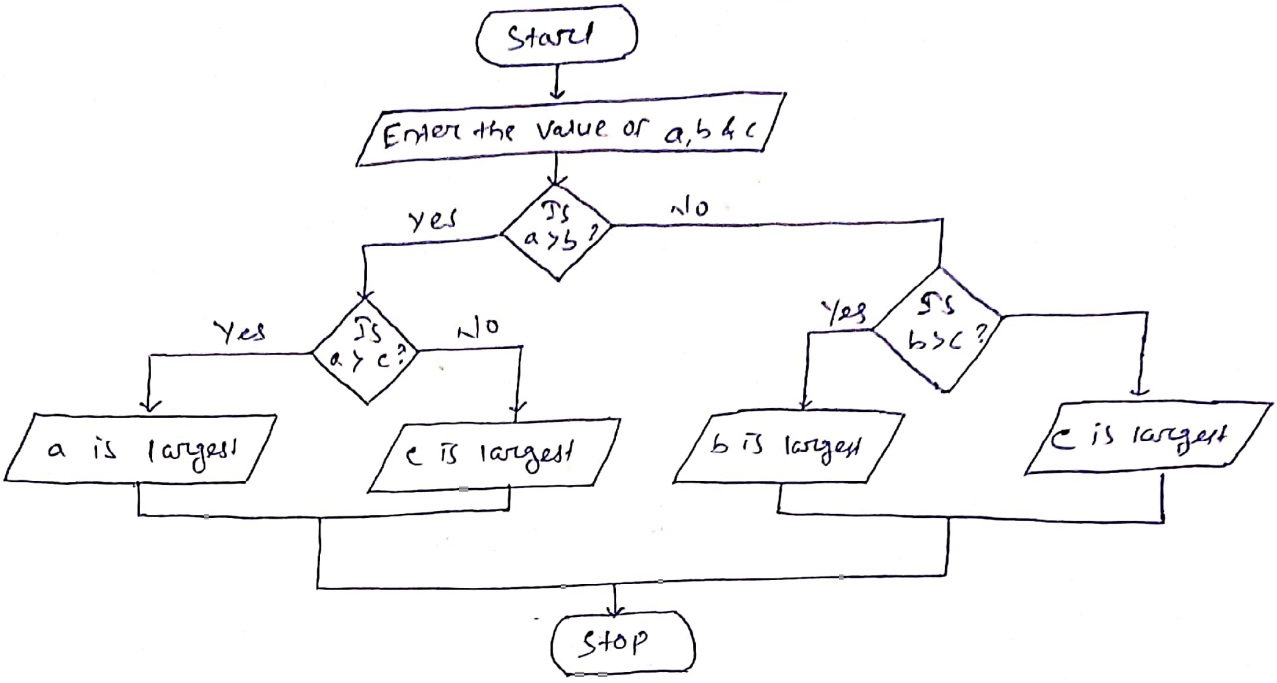
Ex-5: Draw a flowchart to swap the values of two variables without using a temporary variable.



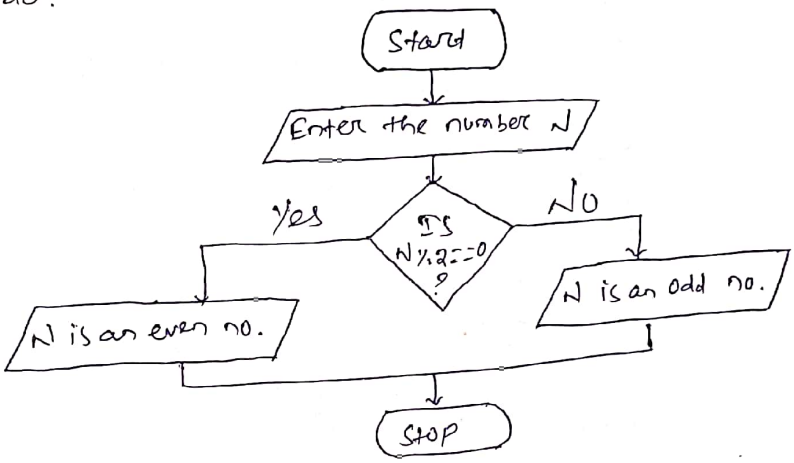
Ex-6: Draw a flowchart to find the largest of two numbers.



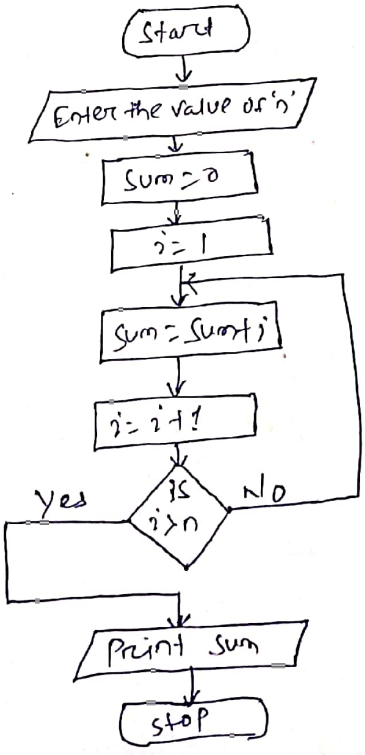
Ex-7: Draw a flowchart to find largest among three numbers.



Ex-8: Draw a flowchard to find whether the given number is even or odd.

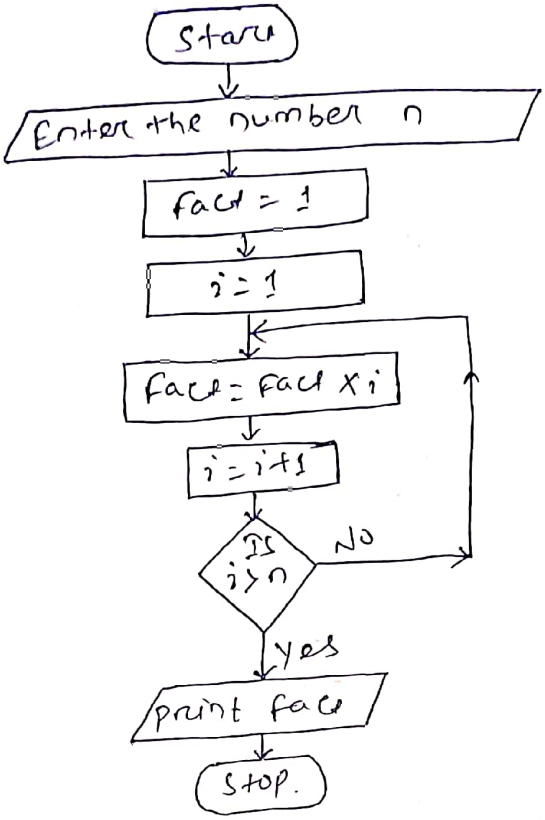


Ex-9: Draw a flowchard to find sum of first n Natural numbers.

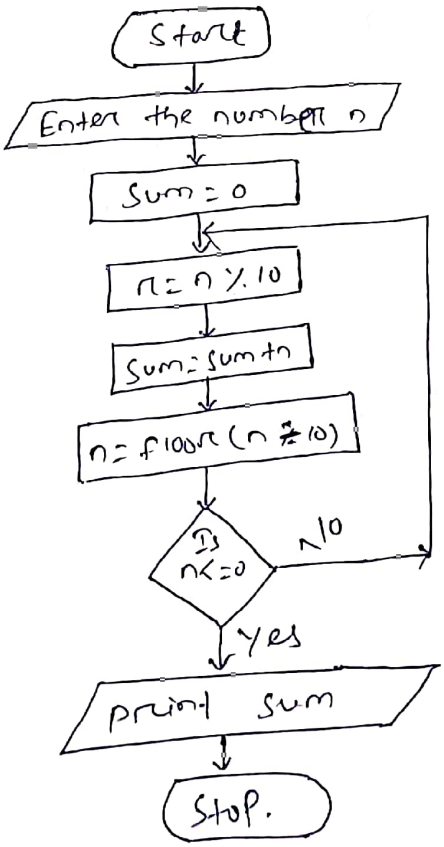


20.10.22

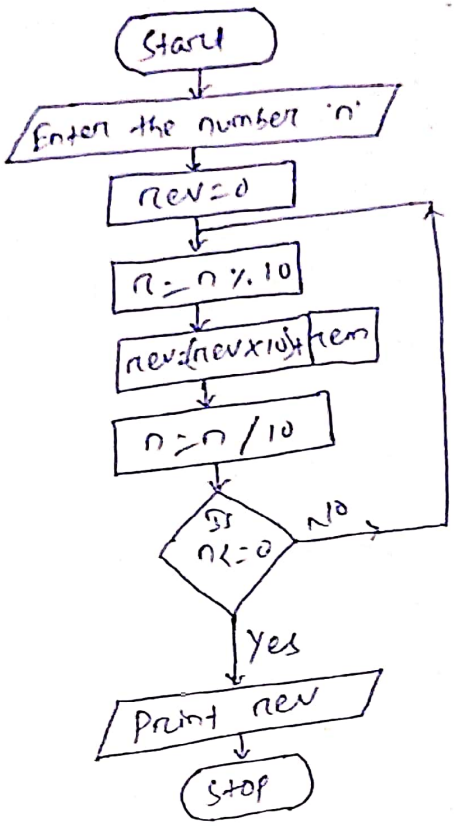
Ex-10: Draw a flowchart to find the factorial of a given number.



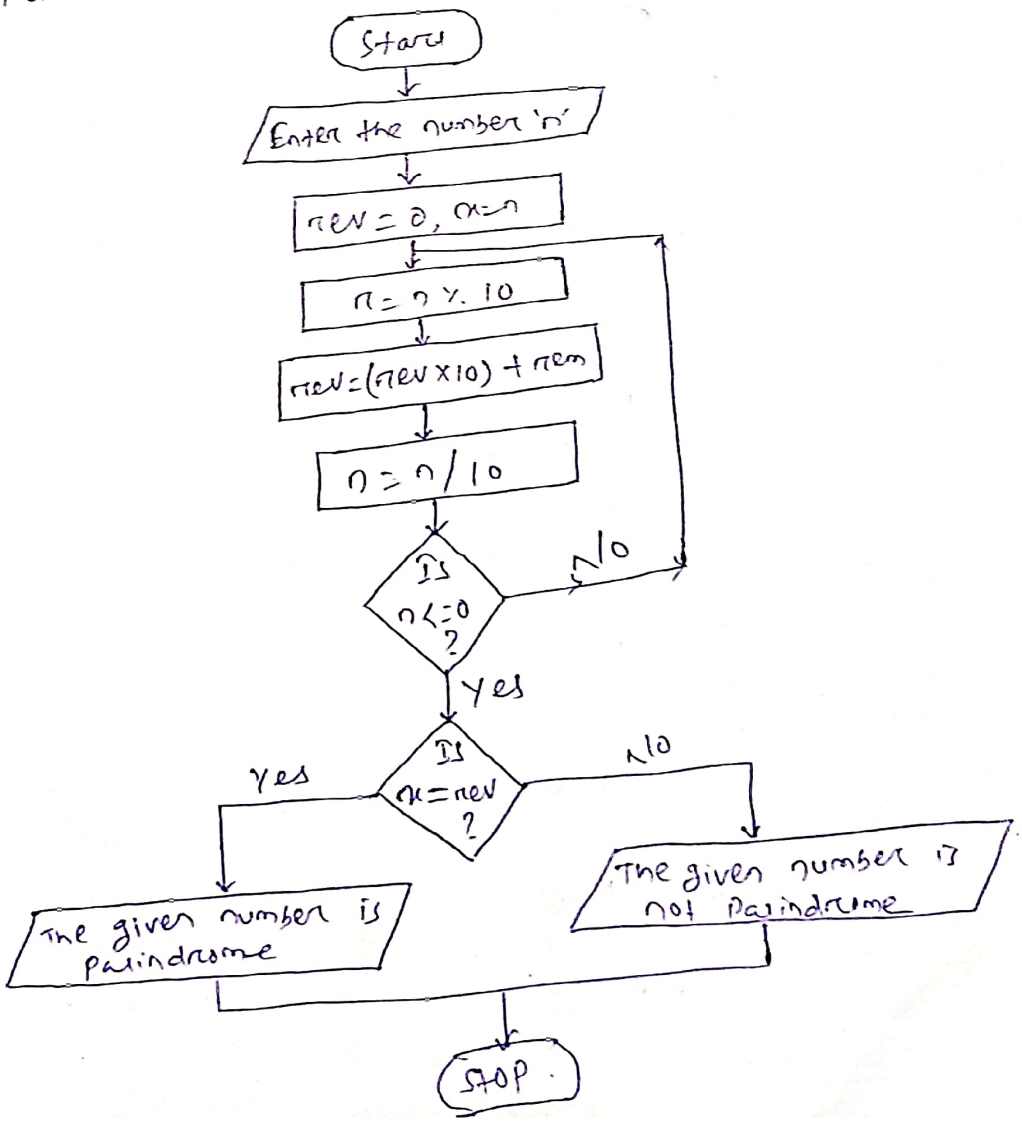
Ex-11: Draw a flowchard to find sum of individual digits of a given number.



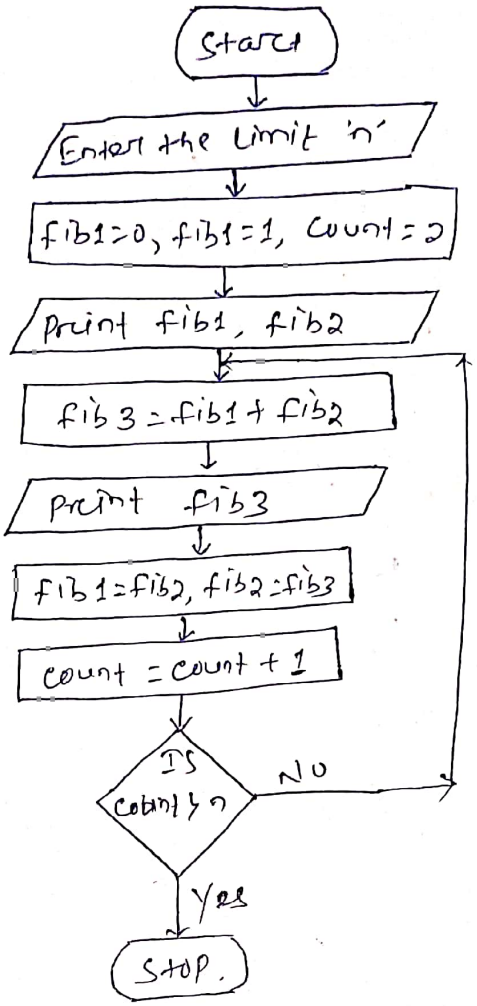
Ex-12: Draw a flowchart to find the reverse of a given number.



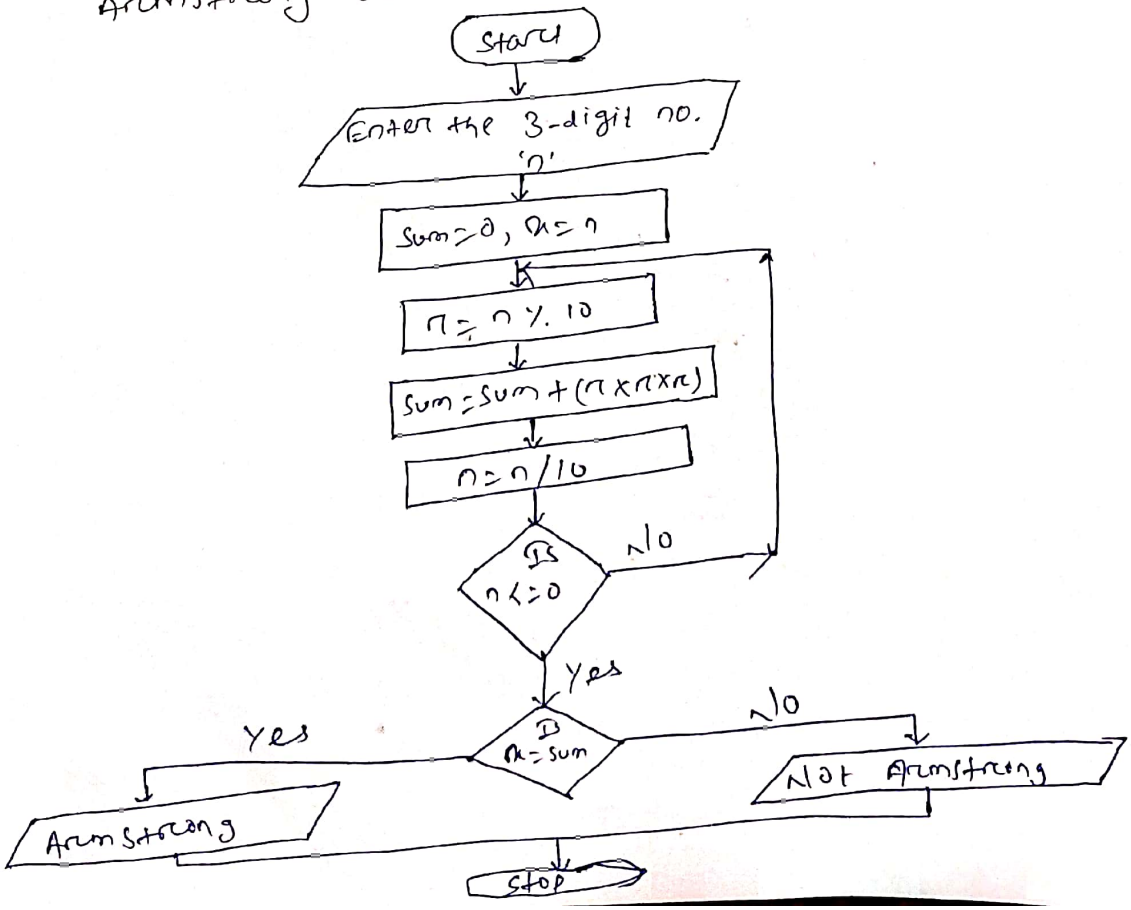
Ex-13: Draw a flowchart to check whether a given number is Palindrome or not.



Ex-14: Draw a flowchart to generate Fibonacci series upto a given limit.



Ex-15: Draw a flowchart to check whether a given 3-digit number is Armstrong or not.



JAVA

26.10.23

- Java was developed by James Gosling in 1995 at Sun Microsystems.
- Later it was acquired by the Oracle company.
- Java is a high level programming language
- Java makes writing, compiling and debugging programming easy.

FEATURES OF JAVA

- **Simple:** Java is one of the simple language as it does not have complex features like pointers and explicit memory allocation (Malloc(), Calloc(), Realloc()).
- **Secure:** Java programs run in an environment that is independent of the operating system environment which makes java programs more secure.
- **Robust:** It is reliable. Checking the errors will be much faster & the java compiler detects those errors that are not easy to detect by another programming language.
  Java is Robust because of the features like Garbage collection, Exception handling and Dynamic Memory Allocation.
- **Distributed:** The java programs can be easily distributed on one or more systems that are connected to each other through an internet connection.
- **Portable:** Java code written on one machine can be run on another machine by carrying the bytecode. This is also called as WORA (Write Once Run Anywhere).
- **Object-oriented:** Java is an Object-oriented programming language. Everything in Java is an Object. Basic concepts of OOPs are: Object, Class, Inheritance, Polymorphism, Abstraction and Encapsulation
- **Machine Independent:** A language that can run on any machine is called Machine Independent language. The JVM can take compiled code for any Java application and run it on any machine.
- **Architectural Neutral:** Java is architecture neutral because there are no implementation dependent features. Example: the size of primitive data types (int, char, double etc.) are almost fixed.
- **Dynamic Flexibility:** Java being completely object-oriented gives us the flexibility to add classes, new methods to existing classes, and even create new classes through sub-classes. Java even supports functions written in other languages such as C, C++ which are referred to as native methods.
- **Multithreaded:** Thread is like a separate program, executing concurrently. We can write Java programs that deal with many tasks at once by defining multiple threads.

→ High performance: Java is faster than other traditional interpreted programming language because Java bytecode is close to native code.

→ Free of cost: Java is a free programming language. We don't need to pay a single rupee for using Java.

APPLICATIONS OF JAVA

→ Desktop based Applications:

-> All desktop applications can easily be developed in Java. Java also provides GUI development capability through various means, mainly Abstract Windowing Toolkit (AWT), Swings and JavaFX.

-> Adobe Creative Cloud is one such which comes as a set of applications and services from Adobe Inc. that gives subscribers access to a collection of software used for graphic design, video editing, web development, photography, along with a set of mobile applications and also some optional cloud services.

→ Web based Applications

-> Java is also used to develop web applications. It provides a vast support for web applications through Servlets, JSPs, Spring and Spring Boot.

-> Most popular applications such as IRCTC, LinkedIn are developed in Java.

→ Mobile based Applications

-> Java is considered as the official programming language for mobile app development.

-> Most popular apps such as Spotify, netflix and twitter were developed using Java.

→ Gaming Applications

-> Java is widely used by game development companies because it has the support of the open-source most powerful 3D engine.

-> The most popular games developed in Java are Minecraft, Mission Impossible III and Asphalt 6 etc.

→ Big Data Applications

-> As many programming languages are available for Big Data Technology but still Java is the first choice for the same.

-> The tool Hadoop platform for processing and storing big data applications is written in Java.

-> In big data, Java is widely used in ETL applications such as Apache Camel, Raptor and Apache Kafka. It is used to extract and transform data, and load in big data environments.

→ Cloud based Applications

-> A cloud application is the on-demand availability of IT resources via the Internet. The cloud-based application provides the service at a low cost.

→ Java provides the environment to develop cloud-based applications.
→ Various Java cloud development tools for cloud computing projects like Oracle Java cloud service, Cloudfoundry, Google App Engine, Openshift, IBM Smart cloud, AWS SDK for Java.

→ IoT based Applications
→ IoT is a technology that connects the devices in its network and communicates with them.
→ IoT has found almost in all the small devices such as health gears, smartphones, wearables, smart lighting, TVs etc.
→ For developing the IoT application there is a lot of programming languages that can be used but Java offers an edge to developers that is unparalleled.
→ Java Embedded Framework provides access from java for hardware and one board computers like Raspberry Pi, Orange Pi, Banana Pi etc. to control SPI/I2C/GPIO or serial ports.

→ AI based Applications
→ Java is being used in many AI applications, from self-driving cars (Ex- XUV 700) to robotic assistants (Robo vaccum cleaners). It is also being used to power big data applications, such as facial recognition systems.
→ Java is also being used to create intelligent chatbots which can interact with users naturally.

→ Scientific Applications
→ Java has enhanced security features which makes it the best option for the development of scientific applications. It has served as a powerfool tool in coding complex mathematical operations.
→ The programs are designed in a highly secure and efficient manner. Some of the most widely used applications like MATLAB use Java as a component of the core system.

→ Software Tools
→ There are many software tools written in Java.
→ For example, Netbeans, IntelliJ IDEA, and Eclipse are all IDEs written and developed in Java.

→ Embedded System
Java is used in some Embedded system like Sim card, BlueRay Discs etc.

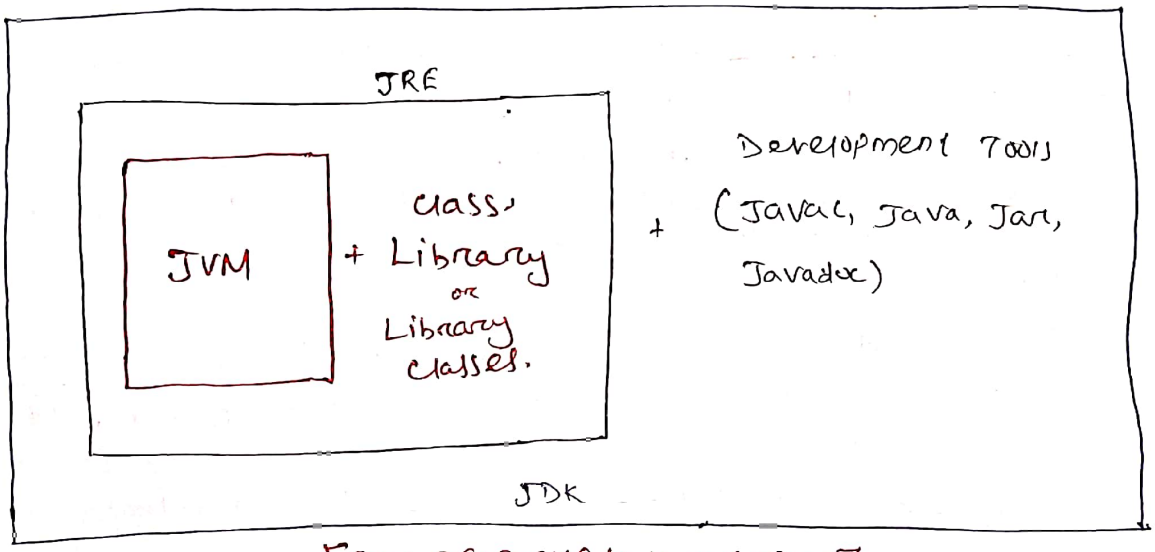
→ Trading
→ Murex is a financial services company that develops software for trading, treasury, risk and post-trade operations in the capital markets.
→ Its software system, also called Murex, is used by over 60,000 daily users in over 60 countries.

JDK SOFTWARE

→ JDK is an acronym for Java Development Kit. The JDK is a software development environment which is used to develop Java applications.

→ JDK is the combination of JRE + Development Tools.

→ The JDK contains a private Java virtual Machine (JVM) and a few other resources such as an interpreter/loader (Java), a compiler (javac), an archiever (jar), a documentation generator (Javadoc) etc. to complete the development of a Java application.



[JAVA DEVELOPMENT KIT (JDK)]

→ JRE is an acronym for Java Runtime Environment. It consists of JVM + Library classes.

→ The JRE contains a JVM and all the class libraries present in the production environment, as well as additional libraries useful to developers.

JDK = JRE + Development Tools
JRE = JVM + Library classes.

jar :→ collection of related java programs.

ARCHITECTURE OF JVM

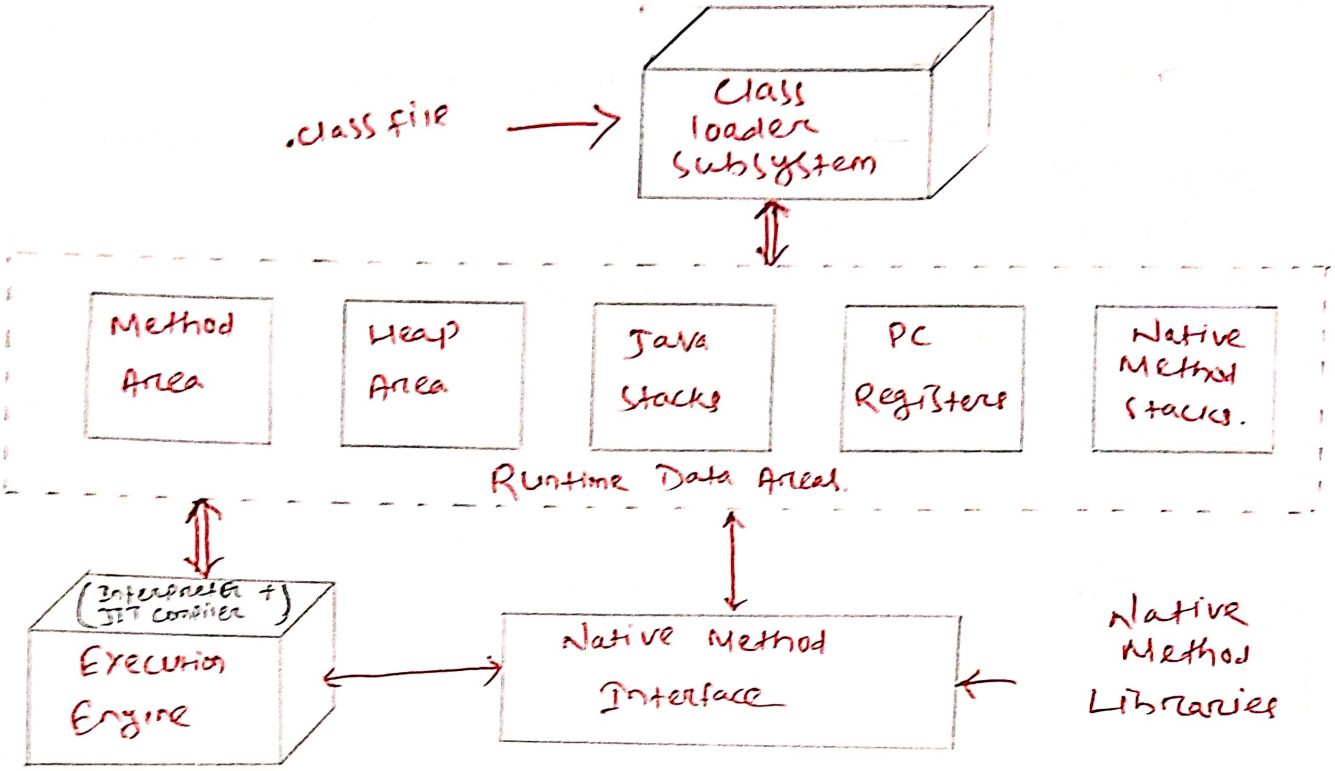
→ JVM stands for Java virtual Machine.

→ JVM is the heard of the entire Java program execution process.

→ First of all we write a program in any editor (notepad, Wordpad etc.) and save the file with .java extension.

→ The .java program is converted into a .class file consisting of byte code instructions by the java compiler at the time of compilation. If there is any error in program then it shows by the compiler & we have to rectify & recompile the .java file & finally we get a .class file. This .class file also called byte code/Intermediate code.

-> The Java compiler is outside the JVM.
-> The .class file is given to the JVM.
Following figure shows the architecture of JVM.



(Fig.- Internal architecture of JVM)

-> In JVM, there is a module (or program) called class loader subsystem, which performs the following instructions

(i) First of all, it loads the .class file into memory.
(ii) Then it verifies whether all byte code instructions are proper or not. If it finds any instruction suspicious, the execution is rejected immediately.
(iii) If byte instructions are proper, then it allocates necessary memory to execute the program.

-> This memory is divided into 5 parts, called run time data areas, which contain the data and results while running the program. These areas are as follows:

Memory Areas:

(1) Method Area
(2) Heap Area
(3) Java Stacks
(4) PC (Program Counter) Registers
(5) Native Method Stacks.

(1) Method Area

Method area is the memory block, which stores the class code, code of the variables and code of the methods in the Java program. (Method means function written in a class).

(2) Heap Area

This is the area where objects are created. Whenever JVM loads a class, method and heap areas are immediately created in it.

(3) Java Stacks

→ Method code is stored on method area. But while running a method, it needs some more memory to store the data and results. This memory is alloted on Java Stacks.

→ So, Java stacks are memory area where Java methods are executed.

→ While executing methods, a separate frame will be created in the Java Stack, where the method is executed.

→ JVM uses a separate thread (or process) to execute each method.

(4) PC Registers

These are the registers (memory areas), which contain memory address of the instructions of the methods.

(5) Native Method Stacks

→ Java methods are executed on Java Stacks. Similarly native methods (for example c/c++ functions) are executed on Native Method Stacks.

→ To execute the native methods, generally native method libraries (for example c/c++ header files) are required. These header files are located and connected to JVM by a program called Native Method Interface.

→ Execution Engine contains interpreter and JIT compiler which translates the byte code instructions into machine language which are executed by microprocessor.

→ Hotspot (loops/iterations in program) is the area in .class file i.e. executed by JIT compiler. JVM will identify the Hotspots in the .class file and it will give it to JIT compiler where the normal instructions and statements of Java programs are executed by the Java Interpreter.

JIT → Just In Time

JDK INSTALLATION & SET UP

→ Download JDK from the oracle website by typing it in any web browser.

→ Then install it.

→ After installation, go to program files → Java → JDK-21(version) → bin. Copy this file path.

→ Type 'env' in search bar and open Edit the System environment variables. Click on environment variables → Path → Edit → new → Paste. Then print OK. Now java path is set in our environment variable.

→ To check whether jdk is set or not, open command prompt and type java and press enter. If it shows that 'java' is not recognized, then jdk path is not set properly. Otherwise it will show some other information.

Write First Program in Java

→ Open notepad and type the program as below.

```
import java.io.*;
class FirstProgram{
    public static void main(String[] args){
        System.out.println("Welcome to JAVA WORLD");
    }
}
```

→ Then save the program as same as classname with .java extension. Here we save as FirstProgram.java.

→ Then go to command prompt and type javac FirstProgram.java. It will give the .class file as FirstProgram.class.

→ Then type java FirstProgram, it will give the output as Welcome to JAVA WORLD.

Describe the program

import java.io.* → To use the Out.println method which belongs to io package.

class → As Java is Pure object oriented program, we have to keep everything inside a class.

FirstProgram → It is the name of the class.

public → It is an access specifier. We should use a public keyword before the main() method so that JVM can identify the execution point of the program. If we use private, protected and default before the main() method, it will not visible to JVM.

Static → The keyword static allows any method to be called without having to instantiate a particular instance of the class. As main() method is called by the Java Interpreter before any objects are made, we need to use static for main() method.

Void → In Java, every method has the return type. If any method does not return any value, then we have to use "void". Here main method does not return anything, so we use Void keyword.

main() → It is a default signature method which is predefined in the JVM. It is called by JVM to execute a program line by line and end the execution after completion of this method. We can also overload the method.

String [] args → The main method also accepts some data from the user. It accepts a group of Strings, which is called a string array. It is used to hold the command line arguments in the form of string values.

Here args[] is the array name, and it is of String type. It means that it can store a group of string. The array can also stores number but in the form of string only. Values passed to the main() method is called arguments. The arguments are stored into args[] so the name args[] is generally used for it.

We can use any names instead of args. but the method signature must not changed.

```
public static void main (String [] a){}
public static void main (String [] des){}
```

We can also interchange public & static, i.e.

```
static public void main (String [] des){}
```

{} → Curly braces are used to open and close the classes as well as methods.

(Cont...)

Q) What happens if the main method is not written properly?

→ The program will compile, but not run, because JVM will not ~~execute~~ recognize the main() method.

→ JVM always looks for the main() method with a string type array as a parameter.

Execution process

→ JVM starts execution from the main method, but if there is any static block, then it executes the static block and then the main() method get executed.

→ JVM executes a static block on the highest priority basis. It means JVM first goes to static block even before it looks for the main() method in the program.

Ex:
```
class Demo{
    static {
        System.out.println("static block");
        }
    public static void main (String [] args){
        System.out.println("Main Method");
        }
}
```

output
static block
Main Method.

→ JVM first search for the static block and executes it and then search for main() method. If main method is not found, it gives error.

```
class Demo{
    static {
        System.out.println("Static block");
    }
}
```

Output

Error: main method not found in class Demo, Please define the main method as public static void main (String [] args) or a JavaFX application must extend javafx.application.Application.

→ So for successful execution of program, every program need a main() method.

→ We can use only 1 main method and 1 public class for a program.

Overloading Main() method

→ We can overload the main() method by making different method signature, but we need the original (predined) main() method in the program to call the overloaded main() methods.

Ex:

```
class Demo{
    public static void main(int a){
        System.out.println("Integer Main");
    }
    public static void main(double d){
        System.out.println("Double Main");
    }
    public static void main(char c){
        System.out.println("Character Main");
    }
    public static void main(String args){
        System.out.println("String Main");
    }
    public static void main(String [] args){
        System.out.println("Original Main");
        main(2);
        main(2.0);
        main('c');
        main("Deb");
    }
}
```

Output

```
Original Main
Integer Main
Double Main
Character Main
String Main
```

Overriding main() method

→ Override is a process where child class can change the implementation of parent class method without changing method signature.

→ The static method can not be override because the static method in java is associated with the class where as non-static method is associated with an object.

→ Static methods don't need an object to be called, static methods can be called directly by using the classname (classname.static_method_name).

→ So, whenever we try to execute the derived class static method, it will automatically execute the base class static method.

→ Therefore it is not possible to override the main method in java as main() method is static.

JAR IN JAVA

→ JAR stands for java archieve.

→ Jar is used to compress all .class files into a single jar file (like a zip file).

→ Jar contains only .class files, not .java files.

→ We can send this jar file to anyone in our network. They can decompress it and decompile it to get the desired code (.java file).

→ We can create a direct executable file by jar.

Methods for creating jar file in CMD

(1) jar cvfe filename.jar javafilename javafilename.class → For create a .jar file.

Ex: jar cvfe first.jar firstProgram firstProgram.class

(2) jar tf filename.jar → To check details of jar file.

Ex: jar tf first.jar.

(3) java -jar filename.jar → To execute the executable jar file.

Ex: java -jar first.jar.

Methods for creating jar file in Eclipse

(1) Click on file → Export → Java → Runnable JAR file. →
(2) Select program in launch configuration.
(3) Select Export destination and name.jar.
(4) Then finish.

→ We can decompile the jar file by a java decompiler tool (Ex:- Java Decompiler → JD-GUI).

(Contd....)

System.out.println():-> In java System.out.println() is a statement which prints the arguments passed to it with a new line.

-> Usually a method is invoked by objectname.methodname().
Print() is the method of PrintStream class, so it should be like

```
PrintStream.obj.print("Hello");
```

-> But we can't create the object to PrintStream class directly as above. So java provided an alternative way to create the object of PrintStream class that is `System.out`.

-> Where System is the class name and is declared as final, The out is an instance of the system class and is of type PrintStream. Its access specifiers are public & final. It is an instance of java.io.PrintStream. When we call the member, a PrintStream class object creates internally.

-> So, we can call the print() method as below:

```
System.out.print();
```

It creates the PrintStream class object. This object, by default, represents the output device i.e. the moniton.

BASIC ELEMENTS OF JAVA

28.10.23

-> Java program contains different types of elements like:

(1) White spaces
(2) comments
(3) Tokens

-> White spaces is nothing but the space we enter from the keyboard.
-> Comments are the sentences used by user/coder to increase readability of the code. Comments are represented by double forward slash (//) and comments are ignored by the compiler.
-> A token is the smallest program element which is recognized by the compiler and which treats them as defined for the compiler.
-> A program is a set of tokens which comprise the following elements.

(1) Identifiers or names
(2) Keywords
(3) Literals
(4) Separators
(5) Operators.

Identifiers:

-> Identifier is the name of variables, methods, classes etc.

Rules for framing Names or Identifiers

-> It should be a single word which contains alphabets a to z, A to Z, digits 0-9, underscore(_).
-> It should not contain White spaces and special symbols.
-> It should not be a keyword of Java.
-> It should not be Boolean literal i.e. true or false.
-> It should not be null literal.
-> It can comprise one or more unicode characters which are characters as well as digits.

→ First letter of an identifier can be an alphabet or it could be underscore (_) but not number.
→ second letter can be a digit or alphabet.
→ Keywords can't be identified
→ Successive underscores (__) are not allowed.

Ex: a_b_c, _java, Java1.

Conventions for writing Names

→ Class: Names of classes and interfaces starts with an upper-case letter. followed by lower case letters.

Method: Names of methods start with a lower-case character and second word first letter must be uppercase.!

Ex: toString(), toBinary() etc.

Package: Names of packages are completely in lower-case letters.

Ex: mypackage, java.lang, util

Variables: Names of variables should start with a lower-case character.

Keywords

→ These are special words defined in Java and represent a set of instruction.
→ The keywords represents groups of instructions for the compiler.
→ These are special tokens that have a predefined meaning and their use is restricted.
→ Keywords can't be used as names of variables, methods, classes or packages.
→ These are written in lower case.
→ Keywords of Java language are as:
abstract, void, for, new etc.

Literals

→ These are values represented by a set of character, digit etc.
→ A literal represents a value which may be primitive type, string type, or null type.
→ The value may be a number (either whole or decimal point number) or a sequence of characters which is called String literal, Boolean type etc.
→ A literal is a constant value.
→ Different types of literals are:

(I) Integer literals
(II) Floating point literals
(III) Boolean literals
(IV) Character literals
(V) String literals
(VI) Null literal

Integer Literals

→ Integer literals are the set of digits without a decimal point.
→ The whole numbers are described by different number systems such as:

(I) decimal numbers
(II) octal numbers
(III) hexadecimal numbers
(IV) binary numbers.

Decimal Numbers:

→ These are sequences of decimal digits which are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

→ Ex: 6, 453, 34789 etc.

Octal Numbers:

→ These are sequences of octal digits which are 0, 1, 2, 3, 4, 5, 6, 7

→ These numbers are preceded by 0.

→ Ex: 04, 07122, 042356 etc.

Hexadecimal Numbers:

→ These are sequences of hexadecimal digits which are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

→ The values from 10 to 15 are represented by A, B, C, D, E, F or a, b, c, d, e, f.

→ The numbers are preceded by 0x or 0X.

→ Ex: 0x56ab, 0X6AF2 etc.

Binary Literal:

→ These are sequences of binary digits that are 0, 1.

→ The numbers are preceded by 0b

→ Ex: 0b011001, 0b101, 0b10000 etc.

Floating-point Literal

→ These are the numbers which have a decimal point.

→ Floating-point literal have a integer part followed by a decimal point followed by a fraction part.

→ Ex: 23.456, 4.67876, -123.456 etc.

Boolean Literal

→ These are the boolean values.

→ It has only two states: true / false.

Character Literal

→ These are the values in characters.

→ An alphabet or a digit within a single quote is treated as a character literal.

→ Ex: 'A', 'H', 'K', 'b', '9' etc.

String Literal

→ These are the strings of characters within a double quotes is called String literal.

→ String Literal can be a word as well as a sentence.

→ Ex: "Deb", "Deb@456", "Debashis Mohapatra", "987" etc.

Null Literal

→ Null is a special literal which represents the absence of an object.

→ There is only one value in Null Literal i.e. null.

Separators

→ Separators are also known as Delimiters which are used for separate identifiers, variables, methods, classes etc.

→ Ex: , ; : {} [] () etc.

DATATYPES

→ The classification of data item is called data type.

→ Java defines eight simple types of data

(i) byte
(ii) short
(iii) int
(iv) long
(v) char
(vi) float
(vii) double
(viii) boolean.

→ These can be put in four groups:

(1) Integer Data types
(2) Float Data types
(3) Character Data types
(4) Boolean Data types

Integer Data Types

→ These data types store integer numbers.

→ There are 4 datatypes to store integer values.

Datatype	Memory size	Range
Byte	1 byte	-128 to 127
Short	2 bytes	-32768 to 32767
int	4 bytes	-2147483648 to 2147483647
long	8 bytes	-9223372036854775808 to 9223372036854775807

→ byte b = 10; short s = 100; int i = 1000; long l = 219876L;

→ To store a value in long datatype, we have to write 'L' after the number, otherwise it treated as an ~~Integ~~ int type.

Ex: long l = 2198765L;

Float Data Types

→ These data types store floating point numbers.

→ There are 2 datatypes to store floating point values.

Datatype	Memory size	Range
Float	4 bytes	-3.4e38 to 3.4e38
Double	8 bytes	-1.7e308 to 1.7e308.

e → Exponential Mantissa.

$0.00012 = 1.2 \times 10^{-4} = 1.2e-4$

→ Floating point number always gives 6 digits of accuracy after decimal point.

Ex: 26.547123f;

→ For storing a decimal point value in float, we have to add 'f' after the ending of number, otherwise it treated as double by the compiler.

Ex: float f = 26.547123f;

→ Double gives 16 digits of accuracy after decimal point.

Character Data Type

→ This data type represents a single character within a single quote.

→ All character elements will going to have positive values only.

char → 2 bytes → 0 to 65535

→ Ex: char c='a'; char c='9';

Boolean Data Type

→ This data type can handle truth values either true or false.

→ It has no specific size.

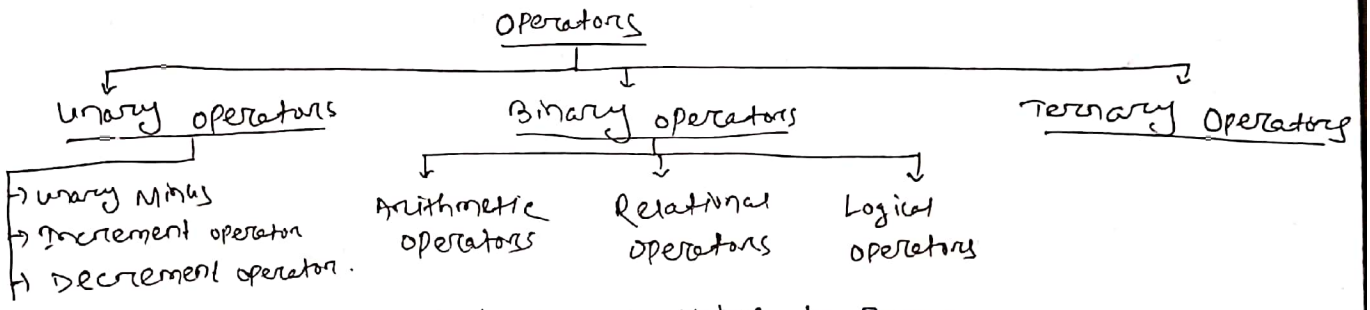
→ Ex: boolean response = false;

Operators

30.10.23

→ An operator is a symbol that performs an operation.

→ An operator acts on variables called operands.



→ Every operator must have two things to follow

(1) Operator precedence (priority)

(2) Operator Associativity (operating direction / Execution direction)

Unary Operator

→ As the name indicates, unary operators act only on one operand.

→ Unary operator is further classified into

(i) Unary Minus operator → R to L

(ii) Increment operator } → L to R

(iii) Decrement operator. }

Unary Minus Operator

→ Unary minus makes positive value as negative. (Associativity from R to L)

Ex: int j = 2;

int k = -j; k = -2;

Increment Operator

→ Increment operator increase the value of a variable by '1'.

→ Increment operator is of 2 types.

(i) Pre Increment operator
(ii) Post Increment operator

Pre Increment operator

→ Pre increment operator will increase the value first & then use it for operation.

→ Syntax : ++variable ;

```
Ex:  int a = 6;
     ++a; // a = 7
     Print (a); → 7
```

Post Increment operator

→ In post increment operator, the value used first & then increased.

→ Syntax : variable++ ;

```
Ex:  int a = 6;
     a++ ; // a = 6
     Print (a); → 7
```

Decrement Operator

→ Decrement operator decrease the value of a variable by '1'.

→ Decrement operator is of 2 types

(i) Pre decrement operator
(ii) Post decrement operator.

Pre decrement operator

→ Pre decrement operator will decrease the value first and then use it for operation.

→ Syntax : --Variable;

```
Ex:  int a = 6;
     --a; // a = 5
     Print (a); → 5
```

Post decrement operator

→ In post decrement operator, the value used first for operation and then decreased.

→ Syntax : Variable--;

```
Ex:  int a = 6;
     a--; // a = 6
     print (a); → 5
```

① int a=5, b=7; int c = --a + b--;

c = --a + b--
= --5 + 7--
= 4 + 7
⇒ c = 11

a = 4
b = 6
c = 11

② int i=10, j, k; j = i++; print (i, j);
k = ++i; print (i, k);

j = i++
= 10++
j = 10

(i, j) = (11, 10)

k = ++i
= ++11
k = 12

(i, k) = (12, 12)

③ int i=10, j, k; j = i--; print (i, j);
k = --i; print (i, k);

j = i--
= 10--
j = 10

(i = 9, j = 10)

k = --i
= --9
k = 8

(i = 8, k = 8)

④ int i=10, j, k, l; j = i++ + i++; print (i, j); → (12, 21)
k = ++i + ++i; print (i, k); → (14, 27)
l = ++i + i--; print (i, l); → (14, 30)

j = i++ + i++
= 10 + 11
= 21
i = 12, j = 21

k = ++i + ++i
= 13 + 14
= 27
i = 14, k = 27

l = ++i + i--
= 15 + 15
= 30
i = 14, l = 30

⑤ int x=10, y; y = ++x + x++ + x++; print (x, y); → (13, 34)

y = ++x + x++ + x++
= 11 + 11 + 12
= 34

(x, y = 13, 34)

⑥ int i=4, x; x = ++i + ++i + ++i; print (i, x); → (7, 18)

x = 5 + 6 + 7 = 18
i = 7, x = 18

⑦ int i=4; i = i++ + i++ + i++; print i; → 15

i = 4 + 5 + 6 = 15

NOTE: Whenever we do perform increment/decrement operation and assign the value to same variable, then no left out operation will be performed on the assigned value.

(8) int i=0; i = i++ - --i + ++i - i--; print i; → 0

i = 0 - 0 + 1 - 1 = 0

31.10.23

(9) int x=2, y=3, S1, S2; S1 = x + (++y); S2 = ++x + y++; print (S1, y); print (S1, S2);

S1 = 2+4 = 6 S2 = 3+4 = 7 (S1, S2) = (6, 7)

(x=2, y=6) (x=3, y=5)

(10) int i=11; i = i++ + ++i; print i; → 24

i = 11 + 13 = 24

(11) int a=11, b=22, c; c = a+b+ a++ + b++ + ++a + ++b; print (a, b, c);

= 11 + 22 + 11 + 22 + 13 + 24 = 103 (13, 24, 103)

a = 13, b = 24, c = 103

(12) int i=1, j=2, k=3, m; m = i-- - j-- - k--; print (i, j, k, m); → (0, 1, 2, -4)

m = 1 - 2 - 3 = -4

i = 0, j = 1, k = 2

(13) int a=1, b=2, c; c = --b - ++a + ++b - --a; print (a, b, c); → (2, 1, 0)

c = 1 - 2 + 2 - 1 = 0

a = 2, b = 1

(14) int i=19, j=29, k;

k = i-- - i++ + --j - ++j + --i - j-- + ++i - j++; print (i, j, k); → (19, 29, -20)

= 19 - 18 + 28 - 29 + 18 - 29 + 19 - 28 = 38 - 58

= -20

i = 19, j = 29, k = -20

(15) int a=12, b=13, c=11;

a = a++ + --b + c++; = 12 + 12 + 11 = 35 a, b, c = (35, 12, 12)

b = b++ + ++a * 2; = 12 + 36 * 2 = 84 a = 36, b = 84, c = 12

c = c++ + a++ * (++b); = 12 + 36 * 85 = 3072 a = 37, b = 85, c = 3072

print (a, b, c); → (37, 85, 3072)

Binary Operators

→ Binary operators works on two operands.

→ Binary operators are classified into

(I) Arithmetic operators
(II) Relational operators
(III) Logical operators.

Arithmetic operators

→ These operators are used to perform fundamental operations like addition, subtraction, multiplication, division etc,

Operator	Meaning	Example	Result
+	Addition	3+4	7
-	Subtraction	5-7	-2
*	Multiplication	5*5	25
/	Division (Gives Quotient)	14/7	2
%	Modulus (Gives Remainder)	20%7	6

⇒ Among the above operators, *, %, / are having first priority and +, - are having the second priority.

⇒ All the operators are evaluated from left to right

Mathematical Expression to Java Expression

Mathematical Expression	Java Expression
$a+b$	a+b
$\frac{x \times y}{z}$	(x*y)/z
$(2x+1)(3y+z)$	(2*x*1)*(3*y+z)
$\frac{(a+b)^2}{(a-b)^2}$	((a+b)*(a+b))/((a-b)*(a-b)) or Math.pow((a+b),2)/Math.pow((a-b),2)
$\log(\frac{x}{y}+c)$	log(x/y+c)
$e^{\lvert a \rvert}+b$	Math.exp(Math.abs(a))+b
$\sqrt{a^2+b^2}$	Math.sqrt(a*a+b*b) or Math.sqrt(Math.pow(a,2)+Math.pow(b,2))
$\left[\frac{2by}{d+1} - \frac{x}{3(z+y)}\right]$	(2*b*y)/(d+1) - x/3*(z+y)

① $2*((i/3)+4*(j-2))$ where $i=8$, $j=5$.

$= 2*((8/3)+4*(5-2))$

$= 2*(2+4*(5-2))$

$= 2*(2+4*3)$

$= 2*(2*12)$

$= 2*14$

$= 28$ Ans

② g = bia/2 + bij * 4/bij - bij + abc/3 where bij = 2, abc = 2.5

= 2/2 + 2 * 4/2 - 2 + 2.5/3

= 1 + 2 * 4/2 - 2 + 2.5/3

= 1 + 8/2 - 2 + 2.5/3

= 1 + 4 - 2 + 2.5/3

= 1 + 4 - 2 + 0.83

= 5 - 2 + 0.83

= 3 + 0.83

= 3.83 [Ans]

③ S = qui * add/4 - 6/2 + 2/3 * 6/god where qui = 4, add = 2, god = 2

= 4 * 2/4 - 6/2 + 2/3 * 6/2

= ~~4 *~~ 8/4 - 6/2 + 2/3 * 6/2

= 2 - 6/2 + 2/3 * 6/2

= 2 - 3 + 2/3 * 6/2

= 2 - 3 + 0 * 6/2

= 2 - 3 + 0/2

= 2 - 3 + 0

= -1 + 0

= -1 [Ans]

④ on = ink * act/2 + 3/2 * act + 2 + tij where ink = 4, act = 1, tij = 3.2

= 4 * 1/2 + 3/2 * 1 + 2 + 3.2

= 4/2 + 3/2 * 1 + 2 + 3.2

= 2 + 3/2 * 1 + 2 + 3.2

= 2 + 1 * 1 + 2 + 3.2

= 2 + 1 + 2 + 3.2

= 3 + 2 + 3.2

= 5 + 3.2

= 8.2 [Ans]

⑤ S = 1/3 * a/4 - 6/2 + 2/3 * 6/3 where a = 4, j = 3

= 1/3 * 4/4 - 6/2 + 2/3 * 6/3

= 0 * 4/4 - 6/2 + 2/3 * 6/3

= 0/4 - 6/2 + 2/3 * 6/3

= 0 - 6/2 + 2/3 * 6/3

$= 0 - 3 + 2/3 * 6/3$

$= 0 - 3 + 0 * 6/3$

$= 0 - 3 + 0/3$

$= 0 - 3 + 0$

$= -3 + 0$

$= -3$ (Ans)

(6) $i = 2 * 3/4 + 4/4 + 8 - 2 + 5/8$

$= 6/4 + 4/4 + 8 - 2 + 5/8$

$= 1 + 4/4 + 8 - 2 + 5/8$

$= 1 + 1 + 8 - 2 + 5/8$

$= 1 + 1 + 8 - 2 + 0$

$= 2 + 8 - 2 + 0$

$= 10 - 2 + 0$

$= 8 + 0$

$= 8$ (Ans)

(7) $k = 3/2 * 4 + 3/8$

$= 1 * 4 + 3/8$

$= 4 + 3/8$

$= 4 + 0$

$= 4$ (Ans)

Relational Operator

→ These operators are used for comparison purpose.

→ Returns the result in form of true / false.

	Operator	Meaning	Example
2nd priority, R to L	==	Equal	x == 3
	!=	Not equal	x != 3
1st priority, L to R	<	Less than	x < 3
	<=	Less than or equal to	x <= 3
	>	Greater than	x > 3
	>=	Greater than or equal to	x >= 3

→ Among the above operators, <, <=, >, >= having the first priority and evaluated from left to right.

→ ==, != having the second priority and evaluated from right to left.

① int a = 4, b = 5;

a < b = 4 < 5 → True

a > b = 4 > 5 → false

a == b 4 == 5 → false

a != b 4 != 5 → True

② int a = 8, b = 4, c = 5;

boolean d = (a+b) <= (b+c); print (d); d → false

= (8+4) <= 4+5

= 12 <= 9 → false

Logical Operators

→ Logical operators are used to construct compound conditions.

→ A compound condition is a combination of several simple conditions.

Operator	Meaning	Example	Explanation
&&	and operator	if (a>b && a>c) print (yes)	If a value is greater than b and c then print yes.
||	OR operator	if (a==1 || b==1) print (yes)	If either a value is 1 or b value is 1 then print yes
!	not operator	if (!(a==0)) print yes	If a value is not equal to zero then print yes.

→ Logical operators are of 3 types

(i) Logical AND
(ii) Logical OR
(iii) Logical NOT

Logical AND

→ Represented by double ampersand (&&)

→ Returns result in form of true/false.

Operand1	Operand2	operand1 && operand2
T	T	T
T	f	f
f	T	f
f	f	f

→ Operands are nothing but relational operators applied result.

Ex: a=4, b=3;

(a>b) && (a<3);

: (4>3) && (4<3);

: T && f

= f

→ In Logical AND if any one of the operand gives false, then the entire result will be false.

Logical OR

→ Represented by double pipeline (||)

→ Returns result in form of true/false

op1	op2	op1 || op2
T	T	T
T	f	T
f	T	T
f	F	F

In Logical OR, if any one of the operand is true, then the total result will be true.

Ex: a = 4, b = 3;

(a>b) || (a<b);

= (4>3) || (4<3);

= T || F

= T

Logical Not

It reverse the value of the variable.

Ex: boolean b = true;

!b = false

Note:

In a Logical AND (&&) operator if left side relation operator will false then it will not move to right side.

In a Logical OR (||) Operator if left side relation operator will true, then it will not move to right side.

Ex: a = 4, b = 3

(a>b) && (a<b)

F — This won't executed

Return false.

(a<b) || (a>b)

T — This won't executed.

Return true.

Ternary Operator

This operator is called ternary because it acts on 3 variables in the form of expressions.

Syntax:

Variable = ⟨Expression1⟩ ? ⟨Expression2⟩ : ⟨Expression3⟩

First Expression1 is evaluated. If it is true, then expression2 value is stored into variable otherwise Expression3 value is stored into variable.

Ternary operators are evaluated from Right to Left.

Ex: int a, b, max;

max = (a>b) ? a : b

If a>b == true, then max = a

If a>b = false, then max = b

(1) Write a java program to find the largest of two numbers using ternary operator.

```
int a = 37, b = 45; largest
largest = (a>b) ? a : b;
System.out.println(largest); // 45
```

② Write a java program to find the largest of three numbers using Ternary operator.

→
```
int a=4, b=5, c=3, largest;
largest = (a>b) ? ((a>c) ? a : c) : ((b>c) ? b : c);
System.out.println(largest); //5
```
(or)
```
Largest = ((a>b) && (a>c)) ? a : ((b>c) ? b : c);
System.out.println(largest); //5
```

③ Write a java program to check the given number is even or odd using a ternary operator.

→
```
int a=9;
String s;
s = (a%2==0) ? "The given number is even" : "The given number is odd";
System.out.println(s); // The given number is odd.
```

④ Write a java program to check whether a given 3-digit number is Palindrome number or not. 1.11.23

→
```
int n=212;
String s;
s = (n%10 == n%100) ? "Palindrome" : "Not Palindrome";
System.out.println(s); // Palindrome
```

⑤ Write a java program to print 1 is input character is capital otherwise print 0 using ternary operator.

→
```
// Checking by using ASCII values
char c='A';
int result;
result = (c>=65 && c<=90) ? 1 : 0 ;
System.out.println(result); //1
```
(or)
```
// Checking by using character/direct method.
char c='A'
int result;
result = (c>='A' && c<='Z') ? 1 : 0 ;
System.out.println(result); //1
```

Shortcut Assignment operator

→ Some unique compound Assignment operators commonly referred to as Shorthand Assignment operators are provided by Java. Because it offers a quick way to appoint an expression to a variable, it is known as Shorthad or shortcut Assignment operator.

Ex: a = a+7;

We can write it as a += 7;

Simple Assignment	Shortcut Assignment
x = x+y	x += y
x = x-y	x -= y
x = x*y	x *= y
x = x/y	x /= y
x = x%y	x %= y

1) int a = 7, b = 8; b += a+b; b -= a+b;

b += a+b

= b+(a+b) = 8 + (7+8) = 8+15 = 23 (Ans)

b -= a+b

= b-(a+b) = 8 - (7+8) = 8-15 = -7 (Ans)

TYPE CONVERSION / TYPE CASTING

→ The process of converting one data type into an another data type is called type conversion or type casting.

→ The type conversion can be done by either automatically or manually

→ The automatic conversion is done by compiler and manual conversion performed by programmer.

→ There are two types of type casting:

(1) Implicit type conversion / Widening type conversion / Automatic type conversion.

(2) Explicit type conversion / Narrowing type conversion.

Implicit Type conversion

→ Converting a lower data type into a higher one is called implicit type conversion or widening type conversion or Automatic type conversion.

→ It is done automatically by the compiler.

→ It is safe because there is no chance to lose data.

byte → short → int → long → float → double

↕

char

Ex: int a = 10;

long b = a;

print (b); // 10

Explicit type conversion

-> Converting a higher data into a lower one is called Explicit type conversion or narrowing.

-> It is done manually by the programmer.

-> If we do not perform type casting then the compiler reports a compile time error.

-> There is a chance of loss of data.

double -> float -> long -> int -> short -> byte
↕
char

Syntax : (desired_type) variable;

Ex:
```
double d = 3.14567890125;
float f = (float) d;
Print(f); // 3.145679
```

ACCEPTING INPUT FROM KEYBOARD

-> A stream represents flow of data from one place to other place.

-> Streams are of two types in java:
(1) Input streams which are used to accept or receive data.
(2) Output Streams which are used to display or write data.

-> Streams are represented as classes in java.io package.

-> To access input and output streams, java provides a class called System class which consists of some methods for standard input, standard error and standard output.

-> System.in -> This represents InputStream object, which by default represents standard input device i.e. Keyboard.
System.out -> This represents OutputStream object, which by default represents standard output device i.e. Monitor.
System.err -> This field also represents PrintStream object, which by default represents monitor. System.out is used to display normal messages and results whereas System.err is used to display error messages.

-> For accepting input data from keyboard, we can use two process:
(1) By using BufferedReader class
(2) By using Scanner class

Accept input from keyboard using BufferedReader class.

-> While using BufferedReader, whenever we enter an input from Keyboard, then the input is taken in form of byte code (0's & 1's) by the InputStream object (System.in).

-> Then this bytecode is converted to character sequence by the help of InputStreamReader.

Then this character sequence stored into BufferedReader.
As for every input, the procession can't come to take the input because it decrease our system performance. So input stored in the BufferedReader.

BufferedReader has some specific size as 1KB, 2KB, 4KB based on our system.
When the BufferedReader capacity exceeds, then processor collects the data from the BufferedReader.

To use BufferedReader, we can use InputStreamReader that can read data from the keyboard.

```
InputStreamReader obj = new InputStreamReader (System.in);
```

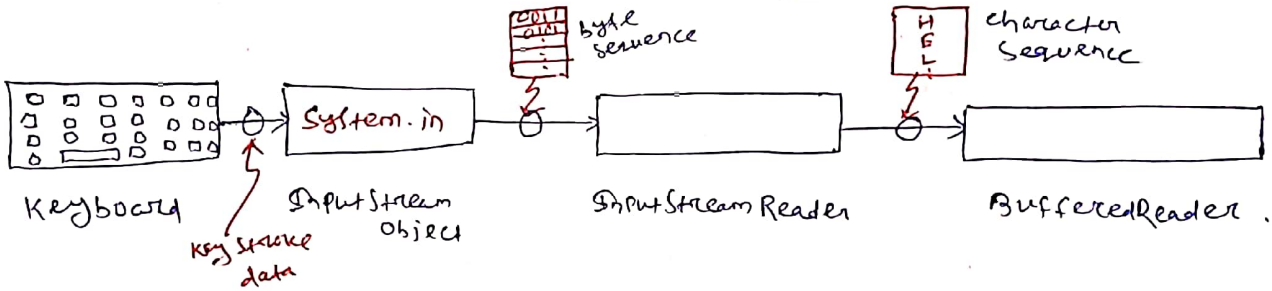
Connect InputStreamReader to BufferedReader, which is another input type of Stream. We are using BufferedReader as it has got methods to read data properly, coming from the stream.

```
BufferedReader br= new BufferedReader (obj);
```

The above two steps can be combined and rewritten in a single line as

```
BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
```

Now we can read the data coming from the keyboard using read() and readLine() methods available in BufferedReader class.



Accepting a single character from the keyboard

Create a BufferedReader class object (br).
For reading a single character from the keyboard, there is a method, but it returns its ASCII number which is an Integer. (br.Read())
Since this integer number can not be stored into character type variable, so we should convert into char type by type Casting method as writing (char) before the method.

```
char ch = (char) br.read();
```

NOTE: Every time we use the above method to read a character and after pressing the enter key, the compiler generates two values as
(i) \n → which is for entering a new line and
(ii) \r → Known as carriage return which is for move the cursor to the beginning of the new line.

If we use BufferedReader to take more inputs after the char data, then we have to skip the above to characters by a method as

```
br.skip(2);
```
Otherwise it gives an error.

Accepting a String from Keyboard

→ Create a BufferedReader class object (br).

→ Then read a string from the keyboard using readLine() method as:

```
String str = br.readLine();
```

→ readLine() method accepts a string from keyboard and returns the string so type casting is not required here.

Accepting an Integer value from Keyboard

→ First we should accept the integer number from the keyboard as a string, using readLine() method as:

```
String str = br.readLine();
```

→ Now the number is in str in form of string. This should be converted into an int by using parseInt() method, a method of Integer class.

```
int n = Integer.parseInt(str);
```

→ We can write the above two statements on a single statement as;

```
int n = Integer.parseInt(br.readLine());
```

→ ParseInt() is a static method in Integer class, so it can be called using class name as Integer.parseInt().

→ We are not using casting to convert string type into int type. The reason is String ~~stats~~ is a class and int is a fundamental data type. Converting a class type into fundamental data type is not possible by using casting. So we are using the method of Integer.parseInt().

Accepting a float value from Keyboard

→ We can accept a float value from the keyboard with the help of the statement as:

```
float f = Float.parseFloat(br.readLine());
```

→ We are accepting the float value in the form of string using readLine() method & then passing the string to Float.parseFloat(), to convert it to float value.

→ ParseFloat() is a static method in Float class.

Accepting a Double value from Keyboard

→ We can accept a double value from the keyboard by using statement

```
double d = Double.parseDouble(br.readLine());
```

→ parseDouble() is a static method in Double class.

Accepting a Byte value from Keyboard

→ We can accept a byte value from the keyboard by using statement

```
byte b = Byte.parseByte(br.readLine());
```

→ parseByte() is a static method of Byte class.

Accepting a short value from keyboard :

→ We can accept a short value by using the statement as

```
Short s = Short.parseShort(br.readLine());
```

→ parseShort() is a static method of Short class.

Accepting a long value from keyboard

→ We can accept a long value by using the statement as

```
Long l = Long.parseLong(br.readLine());
```

Accepting a boolean value from keyboard

→ We can accept a boolean value by using the statement as

```
boolean x = Boolean.parseBoolean(br.readLine());
```

(3) Write a java program which reads various data of a student and print it on the screen using BufferedReader class.

→

```
import java.io.*;
class Buffer
{       public static void main(String[] args){
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
      int rollno, sem;
      String name, branch;
      char gender;
      float marks;
  System.out.println("Enter the roll no.");
  rollno. = Integer.parseInt(br.readLine());
  System.out.println("Enter name");
  name = br.readLine();
  System.out.println("Enter gender");
  gender = (char) br.read();
  br.skip(2);
  System.out.println("Enter branch");
  branch = br.readLine();
  System.out.println("Enter semester");
  sem = Integer.parseInt(br.readLine());
  System.out.println("Enter marks");
  marks = Float.parseFloat(br.readLine());
  System.out.println("Student details");
  System.out.println(rollno + " " + name + " " + gender + " " + branch +
                     " " + sem + " " + marks);
  }
}
```

Accept input from Keyboard using Scanner class

→ The Scanner class is used to get user input.

→ It is available in util package.

→ To use Scanner class, we have to use the import statement as import java.util.*; to get access to methods of Scanner class.

→ To use Scanner class methods to take input from keyboard, first we have to create a Scanner class object.

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

→ We can use this Scanner class reference (sc) to call the methods of Scanner class to take input as:

Read an Integer :	`sc.nextInt();`
Read a floating Number :	`sc.nextFloat();`
Read a character :	`sc.next().charAt(0);`
Read a String :	`sc.nextLine();`
Read a double number :	`sc.nextDouble();`
Read a long number :	`sc.nextLong();`
Read a short number :	`sc.nextShort();`
Read a byte number :	`sc.nextByte();`
Read a boolean value :	`sc.nextBoolean();`

2.11.23

(B.1) Write a java program which reads 3 integer numbers and find their sum and average.

→
```
import java.io.*;
import java.util.*;
class Sum
{
 public static void main (String [] args)
  { int a,b,c, sum; float average;
    Scanner sc = new Scanner (System.in);
    System.out.println("Enter First Number");
    a = sc.nextInt();
    System.out.println("Enter second Number");
    b = sc.nextInt();
    System.out.println("Enter Third Number");
    c = sc.nextInt();
    sum = a+b+c;
    average = sum/3.0f;
    System.out.println("Sum of 3 numbers is: " + sum);
    System.out.println("Average of 3 numbers is: "+ average);
  }
}
```

2) Write a java program to find the area and circumference of the circle by reading the radius of the circle.

→
```
import java.io.*;
import java.util.*;
class Area
{
  public static void main (String [] args)
  {
    float radius, area, circumference;
    Scanner sc = new Scanner (System.in);
    System.out.println("Enter the radius of the circle: ");
    radius = sc.nextFloat();
    area = 3.142f * radius * radius;
    circumference = 2 * 3.142f * radius;
    System.out.println("Area of the circle is: " + area);
    System.out.println("Circumference of circle is: " + circumference);
  }
}
```

3) Write a java program which reads the principle amount, time period, rate of interest from the keyboard and calculate the simple Interest.

→
```
import java.io.*;
import java.util.*;
class SimpleInterest
{
  public static void main (String [] args)
  {
    int p,t;
    float r, si;
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the principal amount");
    p = sc.nextInt();
    System.out.println("Enter the time period");
    t = sc.nextInt();
    System.out.println("Enter the rate of interest");
    r = sc.nextfloat();
    si = p*t*r/100;
    System.out.println("Simple Interest is:" + si);
  }
}
```

④ Write a java program which reads temperature in fahrenheit and convert into celcius.

→
```
import java.io.*;
import java.util.*;
class Temperature
{
  public static void main (String [] args)
  {
    float ft, ct;
    Scanner sc: new Scanner(System.in);
    System.out.println("Enter the temperature in fahrenheit");
    ft = sc.nextFloat ();
    ct = (ft - 32.0f) * 5/9 ;
    System.out.println("celcius temperature is:" + ct);
  }
}
```

⑤ Write a java program which reads two integer numbers and swap these two values using a temporary variable.

→
```
import java.io.*;
import java.util.*;
class Swap
{
  public static void main (String [] args)
  {
    int a, b, temp;
    Scanner sc= new Scanner (System.in);
    System.out.println ("Enter the value of a: ");
    a= sc.nextInt ();
    System.out.println ("Enter the value of b:");
    b= sc.nextInt();
    System.out.println("Before swapping, the value of a= " + a + " the
                        value of b= " +b);

    temp = a;
    a= b;
    b= temp;
    System.out.println("After swapping, the value of a= " + a +
                       "the value of b= " + b );
  }
}
```

(5) Write a java program which reads two integer numbers and swap these two values without using a temporary variable.

→
```
import java.io.*;
import java.util.*;
Class Swap
{
  Public Static void main (String [] args)
  {
    int a, b;
    Scanner sc: new Scanner (System.in);
    System.out.println (" Enter the value of a");
    a: sc.nextInt ();
    System.out.println (" Enter the value of b");
    b: sc.nextInt ();
    System.out.println ("value of a=" +a + "value of b=" +b);
    a = a+b;
    b: a-b;
    a: a-b;
    System.out.println(" After swapping, Value of a=" +a + " Value of
                                    b=" + b);
  }
}
```

COMMAND LINE ARGUMENTS

→ The arguments that we passed during the execution of a program at command prompt is called Command Line Argument.

→ Command-Line arguments in java are used to pass arguments to the main program.

→ If we look at the java main method syntax, it accepts String array as an argument.

→ When we pass command-line arguments, they are treated as Strings and passed to the main function in the String array argument.

→ Ex:
```
Class Commandlinearg
{
  Public Static void main (String [] args)
  {
    System.out.println (args[0] + " " + args[1] + " " + args[2]);
  }
}
```

MATH CLASS Available in java.lang package.

→ Math class in Java has some methods that help to perform numeric operations like square, square root, cube, cube root, exponential, trigonometric operations etc.

→ By Math class methods we can perform mathematical operations and can make long calculations a bit easy.

→ **Methods of Math Class**

(1) **Math.pow()**

→ It is used to calculate the power of a given number.

→ For a^2, we can write Math.pow(a,2);

for a^3, we can write Math.pow(a,3); etc.

(2) **Math.abs()**

→ It is used to calculate the absolute value (positive value) of a given value.

→ For -a, we can write Math.abs(-a); → a

(3) **Math.sqrt()**

→ This method is used to return the square root of a number.

→ Ex: Math.sqrt(16); → 4

(4) **Math.cbrt()**

→ This method is used to return the cube root of a number.

→ Ex: Math.cbrt(27); → 3

(5) **Math.floor()**

→ This method is used to return the nearest lower value of a given decimal or float value.

→ Ex: Math.floor(2.3); → 2.0

(6) **Math.ceil()**

→ This method is used to find the nearest upper value of a given decimal or float value.

→ Ex: Math.floor(2.3); → 3.0

(7) **Math.max()**

→ This method is used to find the maximum number among 2 values.

→ Ex: Math.max(a,b);

(8) **Math.min()**

→ This method is used to find the minimum value among 2 values.

→ Ex: Math.min(a,b);

(9) Math.exp()

→ This method is used to find the exponential value.

→ Ex: Math.exp(4); → 54.598150033144236

(10) Math.log()

→ This method is used to find the logarithm value with base-2.

→ For base-2, we don't need to mention it.

→ Ex: Math.log(4); → 1.3862943611198906

(11) Math.log10()

→ This method is used to find the logarithm value with base-10.

→ For base-10, we have to mention it.

→ Ex: Math.log10(10); → 1.0

(12) Math.random()

→ This method is used to generate random numbers.

→ It generate random numbers between 0 to 1 in form of double.

→ To get integer value we have to explicit type cast it as;

```
(int)(Math.random()*(max-min+1)+min);
```

Where min = minimum range
max = maximum range

The above method will return integer value between min & max.

(13) Math.toRadians()

→ This method is used to convert a degree value to its equivalent radian value.

→ Ex: Math.toRadians(30); → 0.5235987755982988

(14) Math.sin()

→ This method is used to calculate the sine value of a given degree by converting the degree value to its equivalent radian value.

→ Ex: Math.sin(Math.toRadians(30)); → 0.49999999999999994

(15) Math.cos()

→ This method is used to calculate the cosine value of a given degree by converting the degree value to its equivalent radian value

→ Ex: Math.cos(Math.toRadians(30)); → 0.8660254037844387

(16) Math.tan()

→ This method is used to calculate the tan value of a given degree value by converting the degree value to its equivalent radian value.

→ Ex: Math.tan(Math.toRadians(30)); → 0.5773502691896257.

These are some examples of Math class methods.

CONTROL STATEMENT

→ Java compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear.

→ However, java provides statements that can be used to control the flow of Java code. Such statements are called control statements.

→ It is one of the fundamental features of java, which provides a smooth flow of program.

→ Java provides 3 types of control statements.

(1) Conditional Statements
(2) Looping / Iterative / Repetitive statements.
(3) Unconditional Statement.

Conditional Statements:

→ Conditional statements in Java are the executable block of code (or branch to a specific code) dependent on certain conditions. These statements are also known as decision statements or selection statements in Java.

→ Condition statements are following types:

(i) Simple if statement
(ii) If-else statement.
(iii) Nested-if statement
(iv) If-else-if statement
(v) Switch statement

Simple if statement

→ It is used to execute a statement or a set of statements conditionally.

→ Syntax: if (condition)

Beginning of if ← {
    set of statements
end of if ← }
next statements of program;

→ When the condition is true, then it enters into the body of if and executes the statements

→ When the condition is false, then it will executes the statements after the end of if-block.

→ Statements inside if-block can be 2 types:

(i) Simple statements
(ii) compound statements.

Simple statement → only one statement.
Compound Statements → More than one statement.

→ For simple statements, it is not mandatory to put the statement inside curly braces of if-block, but for compound statements, it is mandatory to put the statements inside curly braces

① Write a java program which reads a number and check

→
```
import java.io.*;
import java.util.*;
class OddCheck
{
  public static void main(String[] args)
  {
    int n; Scanner sc = new Scanner(System.in);
    System.out.println("Enter the number:");
    n = sc.nextInt();
    if (n%2 != 0)
      System.out.println(n+" is an odd number");
  }
}
```

② Write a java program which reads two floating point numbers and find their ratio. If the ratio is greater than 0, then swap the values of these two numbers.

→
```
import java.io.*;
import java.util.*;
class RatioSwap
{
  public static void main(String[] args)
  {
    float a, b, ratio, temp;
    System.out.println("Enter first number:");
    a = sc.nextFloat();
    System.out.println("Enter secund number:");
    b = sc.nextFloat();
    ratio = a/b;
    System.out.println("Before If, value of a= "+a+" value of b= "+b);
    if (ratio>0)
    {
      temp = a;
      a = b;
      b = temp;
    }
    System.out.println("Affere If, value of a= "+a+" value of b= "+b);
  }
}
```

If-else Statement

→ If a programmer wants to choose one among the two alternatives, then we use if-else statement.

→ The if-else statement is an extension to the if-statement, which uses another block of code, i.e. else block. The else block is executed if the condition of the if-block is evaluated is false.

→ Syntax:

```
if (condition)
{
  Statement 1; // executes when condition is true
}
else
{ Statement 2; // executes when condition is false
}
```

① Write a java program which reads two integer numbers and check whether they are equal or not.

→
```
import java.io.*;
import java.util.*;
Class Equality
{
  Public static void main (String [] args)
  {
    int a, b;
    Scanner sc = new Scanner (System.in);
    System.out.println ("Enter First Number:");
    a = sc.nextInt();
    System.out.println ("Enter Second Number:");
    b = sc.nextInt();
    if (a == b)
      System.out.println (a+" and "+b+" are equal");
    else
      System.out.println (a+" and "+b+" are unequal");
  }
}
```

② Write a java program which reads an Integer number and check whether its a positive number or negative number. 3.11.23

→
```
import java.io.*;
import java.util.*;
Class Check
{
  Public static void main (String [] args)
  {
```

```
int a;
System.out.println("Enter the number:");
a = sc.nextInt();
if (a>0)
    System.out.println("positive Number");
else
    System.out.println("Negative Number");
}
}
```

③ Write a java program which reads an integer number and check whether it is an even number or odd number.

→
```
import java.io.*;
import java.util.*;
class Evenodd
{
  public static void main(String [] args)
  {
    int n;
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the number");
    n = sc.nextInt();
    if (n%2==0)
      System.out.println(n+" is an Even number");
    else
      System.out.println(n+" is an Odd number");
  }
}
```

④ Write a java program which reads the age of a person and checks whether he is eligible for voting or not.

→
```
import java.io.*;
import java.util.*;
class Votecheck
{
  public static void main(String [] args)
  {
    int age;
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the age of the person");
    age = sc.nextInt();
    if (age >= 18)
      System.out.println("The person is eligible for vote");
    else
      System.out.println("The person is not eligible for vote");
  }
}
```

(5) Write a java program which reads a character from the keyboard and print 'Yes' if the input character is Y or y otherwise print 'No'.

→
```
import java.io.*;
import java.util.*;
class check
{
  public static void main(String [] args)
  {
    char ch;
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the character");
    ch = sc.next().charAt(0);
   // if(c=='Y' || c=='y')  -> By using character
    if(c==89 || c==121)  -> By using ASCII code of characters.
      System.out.println("Yes");
    else
      System.out.println("No");
  }
}
```

(6) Write a java program to check whether the given number is divisible by 5 or not.

→
```
import java.io.*;
import java.util.*;
class Divisible by 5
{
  public static void main(String [] args)
  {
    int n;
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the number");
    n = sc.nextInt();
    if(n%5==0)
      System.out.println(n+" is divisible by 5");
    else
      System.out.println(n+" is not divisible by 5");
  }
}
```

(7) Write a java program to check the given number is divisible by both 2 & 3 or not.

→
```
import java.io.*;
import java.util.*;
```

```
class Divisibilitycheck
{
  public static void main(String [] args)
  {
    int n;
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter the number");
    n= sc.nextInt();
    if (n%2==0 && n%3==0).
      System.out.println(n+" is divisible by both 2 & 3");
    else
      System.out.println(n+" is not divisible by both 2 & 3");
  }
}
```

⑧ Write a java program to check a given year is leap year or not.

→
```
import java.io.*;
import java.util.*;
class Leapyear
{
  public static void main(String [] args)
  {
    int year;
    Scanner sc= new Scanner(System.in);
    System.out.println("Enter the year");
    year= sc.nextInt();
    if(((n%4==0) && (n%100!=0)) || (n%400==0))
      System.out.println(year+" is a leap year");
    else
      System.out.println(year+" is not a leap year");
  }
}
```

⑨ Write a java program which reads the values of a,b&c from the keyboard & add them. after addition check whether it is in the range of 100 to 200 or not.

→
```
import java.io.*; import java.util.*;
class Add
{ public static void main (String [] args)
  {
    int a, b, c, sum;
    Scanner sc= new Scanner(System.in);
    System.out.println("Enter the value of a, b, c");
    a= sc.nextInt();
    b= sc.nextInt();
    c= sc.nextInt();
    sum= a+b+c;
    if (sum>=100 && sum<=200)
      System.out.println(sum+" is in the range of 100 to 200");
    else
      System.out.println(sum+" is not in the range of 100 to 200");
  }
}
```

Nested if-else Statement

→ If there are more than two alternatives to select then we choose the nested if-else statement.

→ Nested if-else statement means an if-else statement inside other if or else blocks. It is similar to an if-else statement but they are defined inside another if-else statement.

→ Syntax:

```
if (condition)
{
    if (condition2)
    {
        statement1;
    }
    else
    {
        statement2;
    }
}
else
{
    if (condition3)
    {
        statement3;
    }
    else
    {
        statement4;
    }
}
```

Outer if. → Inner if-else

Outer else → Inner if-else

① Write a java program which accept a number & find the number is divisible by 5 or not. If not print the nearest number from the given number which is divisible by 5.

→
```
import java.io.*;
import java.util.*;
class check {
    public static void main(String[] args){
    System.out.println("Enter the number:");
    Scanner sc=new Scanner(System.in);
    int n= sc.nextInt();
    int r= n%5;
    if (n%5==0)
        System.out.println(n+" is divisible by 5");
    else{
      System.out.println(n+" is not divisible by 5");
      System.out.print("Nearest number which is divisible by 5 is:");
      if (r>2)
          System.out.println(n-r+5);
      else
          System.out.println(n-r);
    }
}
```

(2) Write a java program to find largest among 3 numbers using nested if-else statement.

->
```
import java.io.*;
import java.util.*;
class Largest{
    Public static void main(String [] args){
      int a, b, c;
      Scanner sc = new Scanner (System.in);
      System.out.println("Enter the values of a, b, c");
      a = sc.nextInt();
      b = sc.nextInt();
      c = sc.nextInt();
    if (a>b){
        if (a>c)
          System.out.println(a+" is largest");
        else
          System.out.println(b+" is largest");
        }
    else{
      if (b>c)
        System.out.println(b+" is largest");
      else
        System.out.println(c+" is largest");
      }
    }
}
```

(3) Write a java program to find largest among 4 numbers using nested if-else.

```
if (a>b){
    if (a>c){
        if (a>d)
          System.out.println(a+" is largest");
        else
          System.out.println(d+" is largest");
    else{
        if (c>d)
          System.out.println(c+" is largest");
        else
          System.out.println(d+" is largest");
        }
else{
    if (b>c){
        if (b>d)
          System.out.println(b+" is largest");
        else
          System.out.println(d+" is largest");
    else{
        if (c>d)
          System.out.println(c+" is largest");
        else
          System.out.println(d+" is largest");
        }}}}
```

If-else-if Statement

→ If a programmer want to choose one among the several alternatives, then he can use if-else-if statement.

→ In this, the if statement is followed by multiple else-if blocks. We can create a decision tree by using these control statements in java in which the blocks, where the condition is true is executed and the rest of the ladder is ignored & not executed.

→ If none of the condition is true, then the last else block is executed, if present.

→ Syntax:

```
if (condition1)
    Statement1;
else if (condition2)
    Statement2;
else if (condition3)
    Statement3;
    :
else if (condition n)
    Statement n;
else
    Statement;
```

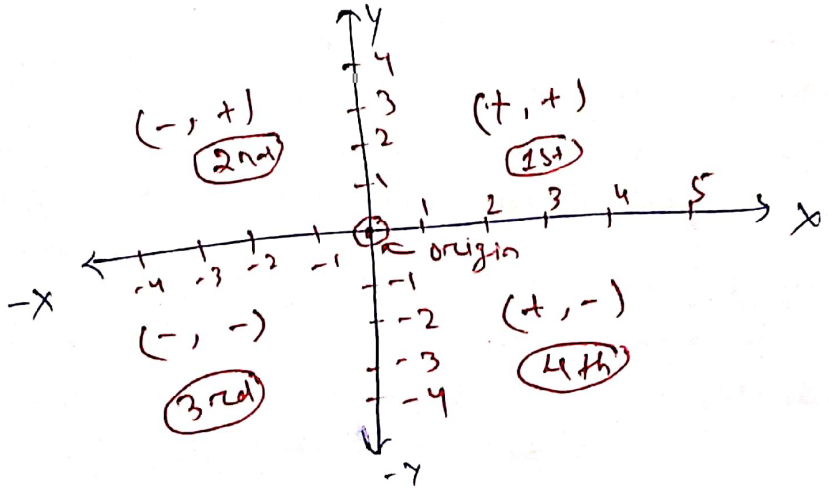
① Write a java program which reads the height of a person in cm and determine whether he is dwarf, average height, taller & abnormal height based on following data:

< 150cm → dwarf
>= 150cm or < 165cm → average height
>= 165cm or < 180cm → taller
> 165cm → abnormal height.

```
System.out.println("Enter the height of the person (in cm)");
float h = sc.nextFloat();
if (h < 150.0f)
    System.out.println("Dwarf");
else if (h >= 150.0f && h < 165.0f)
    System.out.println("Average height");
else if (h >= 165.0f && h < 180.0f)
    System.out.println("Taller");
else
    System.out.println("Abnormal height");
```

(5) Write a java program which reads the x & y values of a coordinate system and determines its quadrant.

```
int x, y;
Scanner sc = new Scanner(System.in);
System.out.println("Enter the value of 'x':");
x = sc.nextInt();
System.out.println("Enter the value of 'y':");
y = sc.nextInt();
if (x>0 && y>0)
    System.out.println("First Quadrant");
else if (x<0 && y>0)
    System.out.println("Second Quadrant");
else if (x<0 && y<0)
    System.out.println("Third Quadrant");
else if (x>0 && y<0)
    System.out.println("Fourth Quadrant");
else if (x==0 && y>0)
    System.out.println("positive side of y-axis");
else if (x==0 && y<0)
    System.out.println("Negative side of y-axis");
else if (x>0 && y==0)
    System.out.println("positive side of x-axis");
else if (x<0 && y==0)
    System.out.println("Negative side of y-axis");
else
    System.out.println("origin");
```



3) Write a java program to calculate an electricity bill by reading the current & previous meter reading. The charges w.r.t. slabs is given as follows:

No. of units consumed	Rate for each unit
0 - 100	0.80
100 - 200	1.20
200 - 300	1.50
> 300	1.80

→

```
float cr, pr, units;
float bill;
Scanner sc = new Scanner(System.in);
System.out.println("Enter current meter reading");
cr = sc.nextInt();
System.out.println("Enter previous meter reading");
pr = sc.nextInt();
units = cr - pr;
if (units>=0 && units<100)
    bill = units * 0.8f;
else if (units>=100 && units<200)
    bill = 80 + (units-100)*1.20f ; // 80 is added for first 100 units
                                    100 is subtracted for first 100 units
else if (units>=200 && units<300)
    bill = (80+120) + (units-200) * 1.50f ; // 80 is added for first 100 units
                                            120 is added for next (100-200) 100 units
else
    bill = (80+120+150) + (units-300) * 1.80f ;
System.out.println("Total bill:" + bill);
```

4) Write a java program to calculate the gross salary of an employee for the conditions given below:

Basic Salary	DA	HRA	Conveyance
>= 5000	110% of BS	20% of BS	500/-
>= 3000 && <5000	100% of BS	15% of BS	300/-
<3000	90% of BS	10% of BS	200/-

Gross Salary = Basic + DA + HRA + conveyance.

→

```
float bs, da, hra, con, gs ;
Scanner sc = new Scanner(System.in);
System.out.println("Enter basic salary");
bs = sc.nextFloat();
```

```
if (bs>=5000)
    {
     da = bs * 110/100;
     hra = bs * 20/100;
     conveyance = 500;
    }
else if (bs>=3000 && bs<5000)
     {
       da = bs;
       hra = bs * 15/100;
       con = 300;
     }
else
     { da = bs * 90/100;
       hra = bs * 10/100;
       con = 200;
     }
 gs = bs+da+hra+con;
 System.out.println("Gross salary = " + gs);
```

(5) Write a java program to find the average of 6 subjects marks of a student and display the result as follows:

avg		Grade
>=35 && <50	→	Third
>=50 && <60	→	Second
>=60 && <70	→	First
>70	→	Distinction.

If any one of the marks of the student is <35, print as fail.

→
```
int s1,s2,s3,s4,s5,s6,total;
float avg;
Scanner sc = new Scanner(System.in);
System.out.println("Enter marks");
s1 = sc.nextInt();
s2 = sc.nextInt();
s3 = sc.nextInt();
s4 = sc.nextInt();
s5 = sc.nextInt();
s6 = sc.nextInt();
total = s1+s2+s3+s4+s5+s6;
avg = total/6;
```

```
if (s1<35 || s2<35 || s3<35 || s4<35 || s5<35 || s6<35)
        System.out.println("Student is fail");
else
    {
        System.out.println("Student is Pass in all Subject");
        System.out.println("Total mark= " + total);
        System.out.println("Average Mark=" + avg);
        System.out.println("Grade: ");
        if (avg>=35 && avg<50)
            S.o.p.("Third");
        else if (avg>=50 && avg<60)
            S.o.p.("second");
        else if (avg>=60 && avg<70)
            S.o.p.("First");
        else
            S.o.p.("Distinction");
    }
```

4.11.23

(6) Write a java program to find the largest of 6 numbers.

-)

```
int a, b, c, d, e, f, largest;
Scanner sc= new Scanner(System.in);
a = sc.nextInt();
b = sc.nextInt();
c = sc.nextInt();
d = sc.nextInt();
e = sc.nextInt();
f = sc.nextInt();
if (a>b)
    largest=a;
else
    largest = b;
if (c>largest)
    largest=c;
if (d>largest)
    largest=d;
if (e>largest)
    largest=e;
if (f>largest)
    largest=f;
System.out.println("Largest number is:" + largest);
```

2) A cloth showroom has announced the following festival discounts on the purchase of items based on the total cost of the item purchased.

Total cost	Discount (in percentage)
< 2000	5%
2000 to 5000	25%
5000 to 10000	35%
> 10000	50%

Write a program to take the input as total cost and compute & display the amount to be paid by the customer after availing the discount.

→
```
float total, disc, pamt;
Scanner sc = new Scanner (System.in);
System.out.println("Enter total cost");
total = sc.nextFloat();
if (total < 2000)
    disc = total * 5/100;
else if (total >= 2000 && total < 5000)
    disc = total * 25/100;
else if (total >= 5000 && total < 10000)
    disc = total * 35/100;
else
    disc = total * 50/100;
pamt = total - disc;
S.O.P.("Total amount = " + total);
S.O.P.("Amount to be paid = " + pamt);
```

3) A library charges fine for books returned late. Following are the fines:

First five days : 40 paise per day
Six to ten days : 65 paise per day
Above ten days : 80 paise per day

Design a program to calculate the fine assuming that a book is returned N days late.

```
int days;
float fine;
days = sc.nextInt();
```

Switch Statement

→ It is also called as Menu Driven Programming.

→ Switch statements are almost similar to the if-else-if statements in java. It is a multi-branch statement.

→ The switch statements have an expression and based on the output of the expression, one or more blocks of codes are executed.

→ These blocks are called as cases. We may provide a default block of code that can be executed when none of the cases are matched similar to the else block.

→ Syntax:

```
switch (expression)
{
  case label1 : Statement1;
                break;
  case label2 : Statement2;
                break;
        :
  case labeln : Statement3;
                break;
  default : Statement
}
```

Rules to be followed:

→ Expression can be variable (int, char, String), Constant (a+5+c), expression etc.

→ Break statement for every case. It is not mandatory to have break in each case. If we do not provide a break statement, the following blocks will also be executed irrespective of the case value. This is known as Trailing case.

→ Duplicate case values are not allowed.

→ Default statement is optional.

→ Statements as compound statement need not placed inside {}

→ Sequence of case label is not required.

→ Case label of variables + constant (a+2) is allowed but constant + constant (1+2) is not allowed.

→ Anything inside starting of switch & above case will ignored by the compiler.

→ Multiple case labels with same type can clubbed together with a single statement.

Ex:
```
case 'A':
case 'a' : S.o.p.("I am A");
```

① Write a java program to implement a simple calculator using switch stmt.

→
```
int a, b, add, sub, mul, div, mod;
char choice;
S.o.p. ("+: addition"); S.o.p.("-: subtraction");
S.o.p. ("*: multiplication"); S.o.p.("/: division"); S.o.p("%: Modulus");
a = sc.nextInt();
b = sc.nextInt();
choice = sc.next().charAt(0);
switch (choice)
{
case '+': add = a+b;
          S.o.p.("Addition = " + add);
          break;
case '-': sub = a-b;
          S.o.p.("subtraction = "+sub);
          break;
case '*': mul = a*b;
          S.o.p.("Multiplication=" + mul);
          break;
case '/': div = a/b;
          S.o.p.("Division = " + div);
          break;
case '%': mod = a%b;
          S.o.p.("Modulus = " + mod);
          break;
default : S.o.p.("Invalid choice");
}
```

② Write a java program to check whether the entered character is a vowel or not.

```
char ch = sc.next().charAt(0);
switch (ch)
{
case 'A':
case 'E':
case 'I':
case 'O':
case 'U':
case 'a':
case 'e':
case 'o':
case 'i':
case 'u': S.o.p.("It is a vowel");
          break;
default : S.o.p.("It is a consonant");
}
```

③ Write a java program to read the grade of a student and print the following message according to grade

S, S → Excellent
a, A → very Good
b, B → Good
c, C → poor
y, Y → Absent
f, F → Fail.

→
```
char grade = sc.next().charAt(0);
switch (grade)
  {
    case 's':
    case 'S': S.O.P. ("Excellent");
              break;
    case 'a';
    case 'A': S.O.P. ("very good");
              break;
    case 'b': S.O.P
    case 'B': S.O.P. ("Good");
              break;
    case 'c';
    case 'C': S.O.P. ("poor");
              break;
    case 'y':
    case 'Y': S.O.P. ("Absent");
              break;
    case 'f':
    case 'F': S.O.P. ("Fail");
              break;
    default: S.O.P. ("Invalid Grade");
  }
```

④ Write a java program which reads a temperature in either of 3 scales (celcius, fahrenheit, Kelvin) and then print equivalent temperature in other two scales.

→
```
S.O.P. ("F: fahrenheit");
S.O.P. ("C: celcius");
S.O.P. ("K: Kelvin");
S.O.P. ("Enter your choice:");
char choice = sc.next().charAt(0);
float temp, ch Kt, ft;   temp = sc.nextFloat();
```

```
Switch (choice)
    {
      case 'f' : ft = temp;
                 ct = (ft - 32) * 5/9 ;
                 kt = ct + 273.03f ;
                 break;
      case 'c' : ct = temp;
                 ft = (ct * 9)/5 + 32;
                 kt = ct + 273.03f;
                 break;
      char 'k' : kt = temp:
                 ~~ft = (ct * 9)~~
                 ct = kt - 273.03f ;
                 ft = (ct * 9)/5 + 32;
      default : S.o.p. ("Invalid choice");
    }
     S.o.p. ("Kelvin temp. = " + kt);
     S.o.p. ("Celcius temp = " + ct);
     S.o.p. ("Fahrenheit temp = " + ft);
```

(5) Write a java program to find the area of different geometrical shapes such as circle, square, rectangle, triangle by passing figure code by ch

→

```
float radius, area, length, breadth, height;
  S.o.P. ("001 : circle");
  S.o.P. ("002 : rectangle");
  S.o.P. ("003 : square");
  S.o.P. ("004 : triangle");
  S.o.P. ("Enter the shape code");
  int code = sc.nextInt();
  Switch (code)
      {
        case 001 : S.o.P. ("Enter radius of circle");
                   radius = sc.nextFloat();
                   area = 3.142f * radius * radius;
                   S.o.P. ("Area of ~~radius~~ circle = " + area);
                   break;
        case 002 : S.o.P. ("Enter length of rectangle");
                   length = sc.nextFloat();
                   S.o.P. ("Enter breadth of rectangle");
                   breadth = sc.nextFloat();
                   area = length * breadth;
                   S.o.P. ("Area of rectangle = " + area);
                   break;
```

```
case 003: S.o.p ("Enter the side of the square");
          length = sc.nextFloat();
          area = length * length;
          S.o.p ("Area of square = " + area);
          break;
case 004: S.o.p ("Enter the side of the triangle");
          length = sc.nextFloat();
          S.o.p ("Enter the height of the triangle");
          breadth = sc.nextFloat();
          area = length * breadth/2;
          S.o.p ("Area of triangle is: " + area);
          break;
default : S.o.p ("Invalid option");
}
```

Looping Statement:

6.11.23

→ Looping statement is a feature which facilitates the execution of a set of instructions/functions repeatedly while some conditions evaluates to true.

→ Java provides three ways for executing loops. While all the loops are provides basic functionality, they differ in their syntax and condition checking time. They are:

(1) While loop
(2) Do-while loop
(3) For loop.

While Loop

→ A while loop is a control flow statement that allows code to be executed repeatedly based on given boolean condition.

→ The while loop is used when the number of iteration is not known but terminating condition is known.

→ Loop is executed until the given condition evaluates to false.

→ Syntax:

```
while (condition)
{
    Statement;
}
```

Body of while loop.

→ While loop is also called entry-controlled loop or a pre-test loop as the condition is checked before entering the loop. The test condition is checked first and then control goes inside the loop.

① Write a java program to print the first n natural numbers.

→
```
import java.io.*;
import java.util.*;
class FirstNatural
{
  public static void main (String [] args)
  {
    int n, i=1;
    System.out.println("Enter the value of n:");
    n= sc.nextInt();
    System.out.println("first "+n+"natural numbers are:");
    while (i<=n)
    { System.out.print(i+" ");
      i++;
    }
  }
}
```

③ Write a java program to find sum of first 'n' natural numbers.

→
```
int n,i, sum; Scanner sc= new Scanner(System.in);
S.O.P.("Enter the value of n:");
n=sc.nextInt();
sum= 0;
i= 1;
while (i<=n){
      sum += i;
      i++;
}
S.O.P.("Sum of first" + n + " natural numbers is:" + sum);
```

③ Write a java program to find the sum of positive numbers & negative nos. of given 'n' numbers.

→
```
int n,i,nsum,psum,num;
Scanner sc= new Scanner(System.in);
S.O.P.("Enter the value of n");
n= sc.nextInt();
i= 1;
psum= 0;
nsum= 0;
while (i<=n){
   S.O.P.("Enter the number:");
   num=sc.nextInt();
   if (num>0)
       psum += num;
   else
       nsum += num;
  } i++;
```

```
S.O.P. ("Sum of positive numbers:" + psum);
S.O.P. ("Sum of negative numbers:" + nsum);
```

(4) Write a java program to find the sum of positive and negative numbers of given n numbers until a '0' is occured.

```
int i, psum, nsum;
Scanner sc = new Scanner (System.in);
psum=0;
nsum=0;
i=1;
while (i !=0){
    S.O.P. ("Enter the numbers:");
    i= sc. nextInt ();
    if (i>0)
        psum += i;
    else
        nsum += i;
    }
S.O.P. ("Sum of positive numbers = " + psum);
S.O.P. ("Sum of Negative numbers: " + nsum);
```

(OR)

```
int i, n, num, psum, nsum;
psum=0;
nsum=0;
i = 1;
S.O.P. ("Enter the value of n:");
n = sc.nextInt();
while (i<=n){
    S.O.P. ("Enter the number");
    num = sc. nextInt();
    if (num>0)
        psum += num;
    else if (num<0)
        nsum += num;
    else
        break;
    i++;
    }
S.O.P. ("Sum of positive numbers:" + psum);
S.O.P. ("Sum of negative numbers:" + nsum);
```

(5) Write a java program to find the sum of individual digits of a given number.

```
int i, n, sum, rem;
Scanner sc= new Scanner (System.in);
sum=0;
System.out.println("Enter number");
n= sc.nextInt();
i=n;
while (n>0) {
    rem= n%10;
    sum += rem;
    n=n/10;
}
S.o.p. ("sum of digits of "+i+" is:"+sum);
```

(6) Write a java program to find the sum of individual digits of a given number & repeat the sum until you get the single digit.

```
int i, n, sum, rem;
Scanner sc= new scanner (System.in);
n= sc.nextInt();
i=n;
sum=0;
while (n>0) {
    rem= n%10;
    sum += n;
    n=n/10;
    if (sum >10) {
        rem = sum%10;
        sum = sum/10;
        sum += rem;
    }
}
S.o.p. ("sum of digits of "+i+" is:"+sum);
```

(or)

```
while (n>0 || sum>10){
    if (n==0){
        n=sum;
        sum=0;
    }
    rem = n%10;
    sum += rem;
    n=n/10;
}
S.o.p. ("sum of digits of "+i+" is = "+sum);
```

(7) Write a java program to find the reverse of a given number.

→
```
int i, n, rem, rev;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
rev = 0;
i = n;
while (n != 0)
    {
      rem = n % 10;
      rev = rev * 10 + rem;
      n = n / 10;
    }
S.O.P.("Reverse of " + i + " is: " + rev);
```

(8) Write a java program to check whether a given number is palindrome or not.

→
```
int i, n, rem, rev;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
rev = 0;
i = n;
while (n > 0)
    {
      rem = n % 10;
      rev = rev * 10 + rem;
      n = n / 10;
    }
if (i == rev)
    S.O.P.(i + " is a palindrome number");
else
    S.O.P.(i + " is not a palindrome number");
```

(9) Write a java program to check whether the given number is an armstrong number or not.

→
```
int i, n, rem, count, sum, power, count1, temp;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
i = n;
temp = n;
count = 0;
while (n != 0) {
      n = n / 10;
      count++;
    }
```

```
Sum=0;
while (i>0)
    {
        rem= i%10;
        count1=count ;
        Power=1;
        While (count1 >0)
            { power = power * rem;
              count1 -- ;
            }
            Sum= sum+ power;
            i= i/10;
        }
    if (temp== sum)
        System.o.p (temp+" is an Armstrong number:");
    else
        s.o.p. (temp+" is not an Armstrong number");
```

7.11.23

(10) Write a java program to find gcd (Greatest common divisor) of two given numbers.

→
```
int a,b,rem,num,den;
Scanner sc=new Scanner (System.in);
s.o.p. ("Enter two values");
a= sc. nextInt();
b= sc. nextInt();
if (a>b){
    num=a;
    den=b;
  }
else {
    num=b;
    den=a;
  }
rem= num %. den;
while (rem !=0)
    {
      num=den;
      den= rem;
      rem= num %. den;
    }
  s.o.p. ("Greatest common divison of "+a+" and "+b+" is = "+rem);
```

(11) Write a java program to find the factorial of a given number;

→
```
int n, i=1, fact=1
n= sc. nextInt(); if n==0)
while (i<=n) {          fact=1;
    fact= fact * i;
      i++; }
                        s.o.p. ("factorial of "+n+" is = "+
                                fact);
```

(12) Write a java program to print the following Pattern:

n, n-1, n-2, ..., 2, 1, 0, 1, 2, ... n-2, n-1, n

Where 'n' can be any integer number.

→
```
int n, k=1, i;
Scanner sc = new Scanner(System.in);
n = sc.nextInt();
i = n;
while (i>=0)
    {
      S.O.P(i+" ");
      i--;
    }
while (k<=n)
    {
      S.O.P.(k+" ");
      k++;
    }
```
(or)

```
k = 0-n;
n = n+1;
while (k!=n)
    {
      ~~S.O.P.(k+" ");~~
      S.O.P.(Math.abs(k)+" ");
      k++;
    }
```

(13) Write a java program to convert given binary number to decimal.

→
```
int bnum, dnum=0, base=1, rem;
Scanner sc = new Scanner(System.in);
S.O.P.("Enter the binary number");
bnum=sc.nextInt();
while (bnum>0)
    {
      rem = bnum%10;
      dnum = dnum + (rem*base);
      bnum = bnum/10;
      base = base*2;
    }
S.O.P.("Regd. decimal no. is:"+dnum);
```

(14) Write a java program to convert given decimal number to binary number.

→
```
int bnum=0, dnum, base=1, rem;
Scanner sc = new Scanner(System.in);
S.O.P("Enter the decimal number");
dnum = sc.nextInt();
while (dnum>0)
    {
```

```
        rem=dnum%2;
        bnum=bnum+(rem*base);
        dnum=dnum/2;
        base=base*10;
       }
    S.o.p. ("Regd. binary no. is:"+bnum);
```

(13) Write a java program to check whether a given number is prime or not.

→

```
int i=1, n, count=0;
Scanner sc=new Scanner (System.in);
n=sc.nextInt();
while (i<=n)
    {
     if (n%i==0)
        count++;
     i++;
    }
if (count==2)
     S.o.p. ("The number is prime no.");
else
    S.o.p. ("Th no. is non-prime no.");
```

(or)

```
int flag=0;
i=2;
while (i<=n/2)
     {
       if (n%i==0)
        {flag=1;
         break;
        }
     }
     i++;
    }
if (flag==0){
    S.o.p. ("prime no.");
else
    S.o.p. ("Non-prime no.");
```

(16) Write a java program to check the occurrence of a digit in a given number.

→
```
int n, i, flag, rem, temp;
Scanner sc = new Scanner (System.in);
S.O.P. ("Enter the number");
n = sc.nextInt();                    // 456745344
S.O.P. ("Enter the digit you want to check")
i = sc.nextInt();                    // 4
temp = n;
flag = 0;
while (n>0)
 {
   rem = n%10;
    if (rem==i)
        flag++;
    n = n/10;
 }
S.O.P. ("Occurrence of "+i+" in "+temp+" is = "+flag+" times");
```

o/p → 4 times

(17) Write a java program to find the products of the middle digits of a given number. (if i/p = 12345, then o/p = 2x3x4 = 24)

→
```
int n, i, prod = 1, temp, rem;
Scanner sc = new Scanner (System.in);
n = sc.nextInt();                    // → 123456
temp = n;
n = n/10;
while (n>9)
  {
    rem = n%10;
    prod = prod * rem;
    n = n/10;
  }
S.O.P. ("product of middle digits of "+temp+" is = "+prod);
```

o/p = 120

(18) Write a java program to swap the first & last digit of the given number. (i/p → 123456, o/p → 623451)

→
```
int n, i, count = 0, temp, fd = 0, ld = 0, result, temp2;
Scanner sc = new Scanner (System.in);
n = sc.nextInt();
```

```
temp = n;
ld = n%.10;
while (temp > 0)
   { fd = temp %.10;
     temp = temp/10;
     count++;
   }
int digit = count - 1;
result += ld;
result = (int) (ld * Math.pow(10,digit));
result += (int) (n%. Math.pow(10, digit));
result -= ld;
result += fd;
S.O.P. ("After swapping of first & last digit of " + n + " is : " + result
```

(19) Write a java program to find the next largest number which is less than the given number without the digit entered by the user.
(If given number = 2345, user i/p = 4, o/p → 2339)

```
int n, temp, nl, rem, digit;
Scanner sc = new Scanner (System.in);
n = sc.nextInt();
digit = sc.nextInt();
nl = n - 1;
while (nl>0) { boolean flag=false; temp=nl;
while (temp>0) {
        rem=temp%.10;
        if (rem == digit) {
              flag=true;
              break;
            }
           temp=temp/10;
         }
      if (!flag)
          break;
      nl--;
   }
   if (nl>0)
      S.O.P. (nl);
   else
      S.O.P. ("No such number found");
```

(OR)

(Type-a)

```
int n, temp, rem, count=0, largest, power, digit, s=0;
Scanner sc = new Scanner (System.in);
n = sc.nextInt();
digit = sc.nextInt();
temp = n;
temp1 = n;
while (temp>0){
        temp = temp/10;
        count++;
      }
   power = count-1;
   largest = 0;
   while(n>0){
          int q=0, r, power1;
          q = n/(int) Math.pow(10, power);
          if (q==digit){
              power1=0;
              r=0;
              q=q-1;
              q = q*(int) Math.pow(10, power1);
              while (power1 != power){
                    r = r+q*(int) Math.pow(10, power);
                    power1++;
                   }
              s = q+r;
              break;
             }
          else
              s=q*(int) Math.pow(10, power);
          largest = largest + s;
          n = n%(int) Math.pow(10, power);
          power--;
         }
      largest = largest + s;
      S.O.P. (largest);
```

(20) Write a java program to check whether the given number is an ADAM number or not. ($12^2 = 144$, $21^2 = 441$ → Reverse → 144 So 12 is an ADAM number)

→
```
int n, i, rem, rev=0, sq1, sq2;
Scanner sc= new Scanner (System.in);
n=sc.nextInt();
i=n;
while (n!=0){
        rem=n%10;
        rev=rev*10+rem;
        n=n/10;
    }
sq1= i*i;
sq2=rev * rev; rev=0;
while (sq2 !=0){
        rem= sq2%10;
        rev= rev*10+ rem;
        sq2= sq2/10;
    }
if (sq1== rev)
    S.O.P. (i+" is an ADAM number");
else
    S.O.P. (i+" is not an ADAM NUMBER");
```

(21) Write a java program to find largest digit in a given number.

→
```
int n, temp. rem, largest=0;
Scanner sc= new Scanner (System.in);
n=sc.nextInt();
temp=n;
while (temp>0) {
        rem= temp%10;
        temp= temp/10;
        if (rem> largest)
            largest= rem;
    }
S.O.P. ("Largest digit in"+ n+ " is: "+ largest);
```

(32) Write a java program to convert the digits of a given number into words. (i/p → 4567, o/p → four Five six seven):

→
```
int n, temp, rem, rev=0;
Scanner sc= new Scanner (System.in);
n = sc.nextInt();
temp = n;
while (temp > 0) {
        rem = temp % 10;
        rev = rev * 10 + temp;
        temp = temp/10;
    }
    while (rev > 0){ rem = rev % 10;
      switch (rem) {
          case 0: s.o.p. ("zero");
                  break;
          case 1: s.o.p. ("one");
                  break;
          case 2: s.o.p. ("Two");
                  break;
          case 3: s.o.p ("three");
                  break;
          case 4: s.o.p. ("four");
                  break;
          case 5: s.o.p. ("Five");
                  break;
          case-6: s.o.p. ("six");
                  break;
          case-7: s.o.p. ("seven");
                  break;
          case-8: s.o.p. ("Eight");
                  break;
          case-9: s.o.p ("Nine");
                  break;
      }
      rev=rev/10;
    }
```

Do-While Loop

→ It is used to execute a set of statements repeatedly until the condition becomes false.

→ Do-while loop is also called as exit-controlled loop or post-test loop as the condition is checked after entering the loop.

-> The loop get executed first, then if the condition is true, then the loop will repeated until the condition becomes false.

Syntax:

```
do
{
 statements;
}
while (condition);
```

1) Write a java program to find the sum of odd number between 1 to n by using do-while loop.

```
int i, n, sum=0;
Scanner sc= new Scanner (System.in);
n=sc.nextInt();
i=1;
do {
  sum += i;
  i= i+2;
}
while (i<=n);
S.O.P. ("sum of first Odd numbers between 1 to" + n + " is = " + sum);
```

2) Write a java program to generate fibonacci series upto a given range/limit using do-while loop.

```
int i=0, j=1, k=1, sum=0;
n=sc.nextInt();
S.o.print (i + " " + j + " ");
do {
  sum = i+j;
  S.o.print (sum + " ");
  i= j;
  j= sum;
  k++;
}
while (k<n);
```

3) Write a java program to check whether the given number is a prime number or not by using do-while loop.

```
int i=1, n, count=0;
n=sc.nextInt();
do {
  if (n%i==0)
     count ++
  } while (i<=n);
```

```
if (count==2)
    S.o.p. (n+" is a prime number");
else
    S.o.p. (n+" is not a prime number");
```

(or)

```
boolean flag=false;
    i=2;
    do {
        if (n%i==0 & n!=2){
            flag=true;
            break;
        }
        i++;
    }
    while (i<=n/2);
    if (!flag)
        S.o.p. (n+" is a prime number");
    else
        S.o.p. (n+" is not a prime number");
```

For Loop

→ A for loop is a repetition control structure that allows you to efficiently write a loop that needs to be executed a specific number of times.

→ A for loop is useful when you know how many times a task is to be repeated. It is an entry-controlled loop.

→ Syntax

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

Ex:

```
for (int i=1; i<=10; i++)
    {
    S.o.p. (i+" ");
    }
```

→ We can also write the for loop in different ways as follw:

(1)
```
for (i=0, j=10; i<j; i++, j--)
    {
    }
```

(2)
```
for (i=1; i<10; i++)
    {
    ;
    }
```

(3)
```
int i=1;
for (; i<10; )
    {
    i++;
    }
```

(1) Write a java program to print multiplication table for a given number.

→
```
int n, i;
n = sc.nextInt();
for (i=1; i<=10; i++)
    { S.o.p. (n + "*" + i + " = " + (n*i));
    }
```

(2) Write a java program to find the following series. 9.11.23

Sum = 1*2 + 2*3 + 3*4 + ... + n*(n-1);

→
```
int n, i, sum=0;
n=sc.nextInt();
for (i=1; i<=n; i++)
        sum = sum + i*(i+1);
S.o.p. ("sum of the series=" + sum);
```

(3) Sum = 1*2 + 3*4 + 5*6 + ...
```
int n, i, sum=0;
n = sc.nextInt();
for (i=1; i<=2*n; i++)
        sum = sum + (i * ++i);
S.o.p. ("sum of the series=" + sum);
```

④ Sum = 1*3 + 2*4 + 3*5 + ... + n * (n+2).

→
```
int n, i, sum = 0;
n = sc.nextInt();
for (i = 1; i <= n; i++)
    sum = sum + i * (i+2);
S.o.p. ("Sum of the series:" + sum);
```

⑤ Sum = 1 - 2 + 3 - 4 + 5 - 6 + ...

→
```
int n, i, sum = 0, sign = 1;
n = sc.nextInt();
for (i = 1; i <= n; i++)
{
    sum = sum + i * sign;
    sign = sign * (-1);
}
S.o.p. ("Sum of the series:" + sum);
```

⑥ Write a Java program to generate the factors of a given number.

→
```
int n, i;
n = sc.nextInt();
for (i = 1; i <= n; i++)
{
    if (n % i == 0)
        S.o.p. (i + " ");
}
```

⑦ Write a java program to check whether the given number is perfect or not. (perfect no. means if sum of the factors of the number excluding the no. itself is same as the number. factors of 6 = 1, 2, 3, 6, 1+2+3 = 6 6 is perfect)

→
```
int n, i, sum = 0;
n = sc.nextInt();
for (i = 1; i < n; i++)
{
    if (n % i == 0)
        sum = sum + i;
}
if (sum == n)
    S.o.p. (n + " is a perfect number");
else
    S.o.p. (n + " is not a perfect number");
```

(8) Write a java program to generate prime factors of a given number.

```
int n, i;
for (i=2; i<=n; i++)
{ if (n%i==0)
  { n=n/i;
    S.O.P. (i+" ");
    i=1;
  }
}
```

(or)

```
for (i=2; i<=n; i++)
{ while (n%i==0)
  {
    S.O.P. (i+" ");
    n=n/i;
  }
}
```

(or)

```
for (i=2; i<n; i++){
    while (n%i==0){
        S.O.P. (i+" ");
        n=n/i;
    }
}
if (n>2)
    S.O.P. (i);
```

(9) Write a java program to find LCM of given two numbers using For Loop.

```
int a, b, j, i, lcm=1;
a=sc.nextInt();
b=sc.nextInt();
j = (a>b) ? a : b;
for (i=1; i<=j; i++)
{ lcm = j*i;
  if (lcm%a==0 && lcm%b==0)
      break;
}
S.O.P. (lcm);
```

(or)

```
for (i=1; i<j; i++)
{ if (j%a==0 && j%b==0)
  { lcm = j;
    break;
  }
  j++;
}
S.O.P. (lcm);
```

(19) Write a java program which reads a set of numbers till we enter zero & calculate the average.

→
```
int n,i, j=5, count=1;
float avg=0.0f, sum=0.0f;
for (i=1; i<=j; i++)
{   n=sc.nextInt();
    if (n!=0)
    {  sum=sum+n;
       count++;
    }
    else
        break;
    j++;
}
avg = sum/count;
s.o.p. (avg);
```

Nested for loop

→ When there is one or more for loops inside a for loop, we call it nested for loop in Java.

→ In nested for loop, the inner loop will executes completely whenever the outer loop executes.

→ Syntax:

```
for (initialization; cond^n ; incr/decrement)
{  for (initialization; cond^n ; incr/decrement)
   {
      ==  stmts;
   }
   ==  stmts
}
```

Outer for loop → Inner for loop

Ex:

```
for (int i=1; i<=10; i++)
{  for (int j=1; j<=10; j++)
   {  s.o.p. (j);
   }
   s.o.println (" ");
}
```

① Write a java program to print the prime numbers between 2 and n using a nested for loop. Also print total no. of prime numbers.

→
```
int n,i,j, count=0;
n=sc.nextInt();
boolean flag;
for(i=2; i<=n; i++)
{
    flag=false;
    for(j=2; j<=i/2; j++)
    {
        if(i%j==0)
        {
            flag=true;
            break;
        }
    }
    if(!flag)
    { s.o.p.("i"+" ");
      count++;
    }
}
s.o.p.("Total prime numbers="+count);
```

② Write a java program to print the armstrong numbers between 2 to n using nested for loop.

→
```
int n, i, j, temp, temp2, rem, x, y, count=0;
n=sc.nextInt();
for(i=1; i<=n; i++)
{ temp=i;
  sum=0;
  count=0;
  for(; temp>0;)
  { temp=temp/10;
    count++;
  }
  temp2=i;
  for(;temp2>0;)
  { rem=temp2%10;
    x=1;
    y=count;
    while(y>0)
    { x=x*rem;
      y--;
    }
    sum=sum+x;
    temp2=temp2/10;
  }
  if(i==sum)
    s.o.p.(i+" ");
}
```

(3) Write a java program to find whether the given number is a perfect number or not. (factors sum excluding number is same as number)

6 → 1+2+3=6 6 → perfect no.

→
```
int n, i, sum=0
n = sc.nextInt();
for (i=1; i<n ; i++)
{
    if (n%i==0)
        sum = sum+i;
}
if (sum==n)
    S.O.P. (n+" is a perfect Number");
else
    S.O.P. (n+" is not a perfect Number");
```

(4) Write a java program to check whether the given number is strong number or not.

→
```
int n, i, temp, rem, sum=0, fact;
n = sc.nextInt();   145
temp=n;
for (; temp>0;)
{ rem = temp%10;
  if (rem==0)
     fact = 1;
  else
  {
     fact = 1;
     for (i=1; i<=rem; i++)
          fact = fact * i;
  }
  sum = sum+fact;
  temp = temp/10;
}
if (sum==n)
    S.O.P. (n+" is a strong number");
else
    S.O.P. (n+" is not a strong number");
```

(5) Write a java program to find the palindrome numbers lies between 1 to n ?.

```
int n,i, temp, rem, sum=0, rev, count=0;
n=sc.nextInt();
temp=n;
for(i=1; i<=n; i++)
{
    temp=i;
    rev=0;
    for(; temp>0;)
    { rem=temp%10;
      rev=rev*10+rem;
      temp=temp/10;
    }
      if(rev==i)
      { s.o.p(i+" ");
        count++;
      }
}
s.o.p("Total palindrome numbers between 1 to "+n+" are :"+count);
```

Pattern problems

(6) Write a java program to print the pattern as below

(1)

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

```
int n=sc.nextInt();
for(int i=1; i<=n; i++)
{ for(int j=1; j<=n; j++)
  { s.o.p("*");
  } s.o.println();
```

---

(2)

```
1 1 1 1 1
2 2 2 2 2
3 3 3 3 3
4 4 4 4 4
5 5 5 5 5
```

```
int n=sc.nextInt();
for(int i=1; i<=n; i++)
{ for(int j=1; j<=n; j++)
    s.o.p(i);
  s.o.println();
}
```

---

(3)

```
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
```

```
int n=sc.nextInt();
for(int i=1; i<=n; i++)
{ for(int j=1; j<=n; j++)
    s.o.p(j);
  s.o.println();
}
```

(4)

```
5 5 5 5 5
4 4 4 4 4
3 3 3 3 3
2 2 2 2 2
1 1 1 1 1
```

```
int n=sc.nextInt();
for(int i=n; i>=1; i--)
{ for(int j=1; j<=n; j++)
    S.O.P.(i);
} S.O.println();
```

(5)

```
5 4 3 2 1
5 4 3 2 1
5 4 3 2 1
5 4 3 2 1
5 4 3 2 1
```

```
int n=sc.nextInt();
for(int i=n; i>=1; i--)
{ for(int j=n; j>=1; j--)
    S.O.P.(j);
} S.O.println();
```

(6)

```
A B C D E
A B C D E
A B C D E
A B C D E
A B C D E
```

```
for(int i=65; i<=69; i++)
{ for(int j=65; j<=69; j++)
    S.O.P.((char)j);
  S.O.println();
}
```

(or)

```
char a=sc.next().charAt(0);
sc.nextLine();
char B=sc.next().charAt(0);
sc.nextLine();
for(char i=a; i<=B; i++)
{ for(char j=a; j<=B; j++)
    S.O.P.(j);
  S.O.println();
}
```

(7)

```
E D C B A
E D C B A
E D C B A
E D C B A
E D C B A
```

```
for(int i=69; i>=65; i--)
{ for(int j=69; j>=65; j--)
    S.O.P.((char)j);
  So.println();
}
```

(8)

```
E E E E E
D D D D D
C C C C C
B B B B B
A A A A A
```

```
for(int i=69; i>=65; i--)
{ for(int j=69; j>=65; j--)
    S.O.P.((char)i);
  S.O.println();
}
```

(9)

```
A A A A A
B B B B B
C C C C C
D D D D D
E E E E E
```

```
for (int i=65; i<=69; i++)
{for (int j=65; j<=69; j++)
    S.O.P ((char) i);
 S.O.Pln ();
}
```

(10)

```
Z X X W V
Z Y X W V
Z Y X W V
Z Y X W V
Z Y X W V
```

```
char a = Z, b = V;
for (char i = a; i<=b; i--)
{ for (char j = a; j<=b; j--)
    S.O.P. (j);
  S.O.Pln();
}
```

(11)

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

```
int n = sc.nextInt ();
for (int i=1; i<=n; i++)
{ for (int j=1; j<=i; j++)
    S.O.P. ("*");
  S.O.Pln ();
}
```

(2)

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

```
int n=sc.nextInt ();
for (int i = 1; i<=n; i++)
{ for (int j=1; j<=i; j++)
    S.O.P. (i);
  S.O.Pln ();
}
```

(3)

```
5
5 4
5 4 3
5 4 3 2
5 4 3 2 1
```

```
int n=sc.nextInt ();
for (int i=n; i>=1; i--)
{ for (int j=n; j>=i; j--)
    S.O.P. (j);
  S.O.Pln ();
}
```

(4)

```
5
4 4
3 3 3
2 2 2 2
1 1 1 1 1
```

```
int n = sc.nextInt ();
for (int i=n; i>=1; i--)
{ for (int j=n; j>=i; j--)
    S.O.P. (i);
  S.O.Pln ();
}
```

(15)

```
A
A B
A B C
A B C D
A B C D E
```

```
char a='A', b='E';
for(char i=a; i<=b; i++)
{ for(char j=a; j<=i; j++)
      S.O.P.(j);
  S.O.Pln();
}
```

(16)

```
A
B B
C C C
D D D D
E E E E E
```

```
char a='A', b='E';
for(char i=a, i<=b; i++){
  for(char j=a; j<=i; j++)
      S.O.P.(i);
  S.O.Pln(" ");
}
```

(17)

```
E
D D
C C C
B B B B
A A A A A
```

```
char a='E', b='A';
for(char i=a, i>=b, i--){
  for(char j=a; j>=i; j--)
      S.O.P.(i);
  S.O.Pln();
}
```

(18)

```
E
E D
E D C
E D C B
E D C B A
```

```
char a='E', b='A', j, i;
for(i=a; j>=b; i--){
  for(j=a; j>=i; j--)
      S.O.P.(j)
  S.O.Pln();
}
```

(19)

```
E D C B A
E D C B
E D C
E D
E
```

```
for(char i='A'; i<='E'; i++){
  for(char j='E'; j>=i; j++)
      S.O.P.(j);
  S.O.Pln();
}
```

(20)

```
E D C B A
D C B A
C B A
B A
A
```

```
for(char i='E'; i>='A'; i--){
  for(char j=i; j>='A'; j--)
      S.O.P.(j);
  S.O.Pln();
}
```

(21)

```
A
B A
C B A
D C B A
E D C B A
```

```
for(char i='A'; i<='E'; i++){
  for(char j=i; j>='A'; j--)
      S.O.P.(j);
  S.O.Pln();
}
```

(23)

```
A B C D E
A B C D
A B C
A B
A
```

```
for (char i='E'; i>='A'; i--){
    for (char j='A'; j<=i; j++)
        S.o.p. (j);
    S.o.pln();
}
```

(23)

```
A B C D E
 B C D E
  C D E
   D E
    E
```

```
for (char i='A'; i<='E'; i++){
    for (char j=i; j<='E'; j++)
        S.o.p. (j);
    S.o.pln();
}
```

(24)

```
    E
   D E
  C D E
 B C D E
A B C D E
```

```
for (char i='E'; i<='A'; i--){
    for (char j=i; j<='E'; j++)
        S.o.p. (j);
    S.o.pln();
}
```

(25)

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

```
for (int i=5; i>=1; i--){
    for (int j=i; j>=1; j--)
        S.o.p. ("* ");
    S.o.pln();
}
```

```
for(int i=1; i<=n; i++)
{ for (int j=n; j>=i; j--)
    S.o.p.("* ");
  S.o.pln();
}
```

(26)

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

```
for (int i=1; i<=n; i++){
    for (int j=n-i; j>=1; j--)
        S.o.p. (" ");
    for (int j=1; j<=i; j++)
        S.o.p. ("* ");
    S.o.pln();
}
```

(27)

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

```
for (int i=1; i<=n; i++){
    for (int j=1; j<i; j++)
        S.o.p. (" ");
    for (int j=n; j>=i; j--)
        S.o.p ("* ");
    S.o.pln();
}
```

```
for (int i=n; i>=1; i--){
    for (int j=1; j<=i-1; j++)
        S.o.p. (" ");
    for (int j=i; j>=1; j--)
        S.o.p. ("* ");
    S.o.pln();
}
```

(28)

```
    A
  B B
C C C
D D D D
E E E E E
```

```
for (int i=65; i<=69; i++){
    for (int j=69; j>i; j--)
        S.o.p. (" ");
    for (int j=i; j>=65; j--)
        S.o.p. ((char) i);
    S.o.pln();
}
```

(29)

```
    A
  A B
A B C
A B C D
A B C D E
```

```
for (char i='A'; i<='E'; i++){
    for (char j=i; j<='E'; j++)
        S.o.p.(" ");
    for (char j='A'; j<=i; j++)
        S.o.p.(j);
    S.o.println();
}
```

---

(30)

```
E
D D
C C C
B B B B
A A A A A
```

```
for (char i='E'; i>='A'; i--){
    for (char j='A'; j<i; j++)
        S.o.p.(" ");
    for (char j=i; j<='E'; j++)
        S.o.p.(i);
    S.o.println();
}
```

---

(31)

```
    A
  B A
C B A
D C B A
E D C B A
```

```
for (char i='A'; i<='E'; i++){
    for (char j=i; j<'E'; j++)
        S.o.p.(" ");
    for (char j=i; j>='A'; j--)
        S.o.p(j);
    S.o.println();
}
```

---

(32)

```
    1
  2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

```
for (int i=1; i<=n; i++){
    for (int j=i; j<n; j++)
        S.o.p.(" ");
    for (int j=i; j>=1; j--)  → for (int j=1; j<=i; j++)
        S.o.p.(i);
    S.o.println();
}
```

---

(33)

```
    5
  4 4
3 3 3
2 2 2 2
1 1 1 1 1
```

```
for (int i=n; i>=1; i--){
    for (int j=1; j<n; j++)
        S.o.p(" ");
    for (int j=n; j>=i; j--)   for (int j=i; j<=n; j++)
        S.o.p.(i);
    S.o.println();
}
```

---

(34)

```
    5
  4 5
3 4 5
2 3 4 5
1 2 3 4 5
```

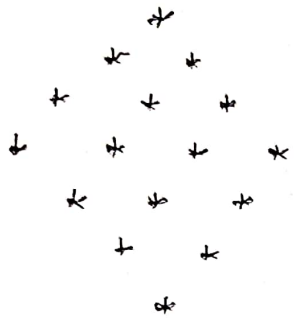
```
for (int i=n; i>=1; i--){
    for (int j=1; j<n; j++)
        S.o.p.(" ");
    for (int j=i; j<=n; j++)
        S.o.p.(j);
    S.o.println();
}
```

(35)

```
    5
  5 4
5 4 3
5 4 3 2
5 4 3 2 1
```

```
for (int i=n; i>=1; i--) {
    for (int j=1; j<i; j++)   for(int j=i; j>1; j--)
        S.o.p.(" ");
    for (int j=n; j>=i; j--)
        S.o.p.(j);
    S.o.pln();
}
```

(36)



```
for (int i=1; i<=n; i++) {
    for (int j=1; j<=n-i; j++)
        S.o.p.(" ");
    for (int j=1; j<=i; j++)
        S.o.p.("*"+" ");
    S.o.pln();
}
for (int i=n-1; i>=1; i--) {
    for (int j=1; j<=n-i; j++)
        S.o.p.(" ");
    for (int j=1; j<=i; j++)
        S.o.p.("*"+" ");
    S.o.pln();
}
```

(37)

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

```
for(int i=1; i<=n; i++) {
    for (int j=1; j<=n-i; j++)
        S.o.p.(" ");
    for (int j=1; j<=i; j++)
        S.o.p.(j+" ");
    S.o.pln();
}
for (int i=n-1; i>=1; i--) {
    for (int j=1; j<=n-i; j++)
        S.o.p.(" ");
    for (int j=1; j<=i; j++)
        S.o.p.(j+" ");
    S.o.pln();
}
```

(38)

```
    5
  4   4
3   3   3
2   2   2   2
1   1   1   1   1
  2   2   2   2
    3   3   3
      4   4
        5
```

```
for (i=n; i>=1; i--) {
    for (j=i-1; j>=1; j--)
        S.o.p.(" ");
    for (j=i; j<=n; j++)
        S.o.p.(i+" ");
    S.o.pln();
}
```

```
for(i=2; i<=n; i++) {
    for (j=1; j<i; j++)
        S.o.p.(" ");
    for (j=i; j<=n; j++)
        S.o.p.(i+" ");
    S.o.pln();
}
```

(9)

```
1
2  3
4  5  6
7  8  9  10
11 12 13 14 15
```

```
int i, j, n, k = 1;
for (i = 1; i <= n; i++) {
    for (j = 1; j <= i; j++)
        S.o.p.(k++);
    S.o.println();
}
```

(10)

```
1
1  0
1  0  1
1  0  1  0
1  0  1  0  1
```

```
for (i = 1; i <= n; i++) {
    for (j = 1; j <= i; j++) {
        if (j % 2 == 0)
            S.o.p.("0");
        else
            S.o.p.("1");
    }
    S.o.println();
}
```

(41)

```
1
0  0
1  1  1
0  0  0  0
1  1  1  1  1
```

```
for (i = 1; i <= n; i++) {
    for (j = 1; j <= i; j++) {
        if (j % 2 == 0)
            S.o.p.("0");
        else S.o.p.("1");
    }
    S.o.println();
}
```

(42)

```
1  0  1  0  1
1  0  1  0
1  0  1
1  0
1
```

```
for (i = 1; i <= n; i++) {
    for (j = n; j >= i; j--) {
        if (j % 2 == 0)
            S.o.p.("0");
        else
            S.o.p.("1");
    }
    S.o.println();
}
```

(43)

```
1  1  1  1  1
0  0  0  0
1  1  1
0  0
1
```

```
for (i = 1; i <= n; i++) {
    for (j = n; j >= i; j--) {
        if (j % 2 == 0)
            S.o.p.("0");
        else
            S.o.p.("1");
    }
    S.o.println();
}
```

44

```
        1
      0 1
    1 0 1
  0 1 0 1
1 0 1 0 1
```

```
for(i=1; i<=n; i++){
    for (j=i; j<n; j++)
        s.o.p(" ");
    for (j=1; j<=i; j++){
        if (j%2==0)
            s.o.p("0");
        else
            s.o.p("1");
    }
    s.o.println();
}
```

45

```
        1
      0 0
    1 1 1
  0 0 0 0
1 1 1 1 1
```

```
for (i=1; i<=n; i++){
    for (j=i; j<n; j++)
        s.o.p(" ");
    for (j=1; j<=i; j++){
        if (i%2==0)
            s.o.p("0");
        else
            s.o.p("1");
    }
    s.o.println();
}
```

46

```
* * * * *
# # # # #
* * * * *
# # # # #
* * * * *
```

```
for(i=1; i<=n; i++){
    for (j=1; j<=n; j++){
        if (i%2==0)
            s.o.p("#");
        else
            s.o.p("*");
    }
    s.o.println();
}
```

47

```
# # # # #
* * * * *
# # # # #
* * * * *
# # # # #
```

```
for(i=1; i<=n; i++){
    for(j=1; j<=n; j++){
        if (i%2==0)
            s.o.p("*");
        else
            s.o.p("#");
    }
    s.o.println();
}
```

48

```
        A
      B   B
    C   C   C
  D   D   D   D
    C   C   C
      B   B
        A
```

```
for (i='A'; i<='D'; i++){
    for (j=i; j<'D'; j++){
        s.o.p(" ");
    for (j='A'; j<=i; j++)
        s.o.p(i);
    s.o.println();
}
for (i='C'; i>='A'; i--){
    for(j='C'; j>=i; j--){ s.o.p(" "); }
    for (j=i; j>='A'; j--){ s.o.p(i); }
    s.o.println();
}
```

(49)

```
D
C   C
B   B   B
A   A   A   A
B   B   B
C   C
D
```

```
for(i='D'; i>'A'; i--){
    for(j=i; j>'A'; j--){
        s.o.p(" ");
    for(j='D'; j>=i; j--)
        s.o.p(j);
    s.o.println();
}
for(i='B'; i<='D'; i++){
    for(j='B'; j<=i; j++)
        s.o.p(" ");
    for(j=i; j<='D'; j++)
        s.o.p(j);
    s.o.println();
}
```

13.11.23

(50)

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

```
for(i=1; i<=n; i++){
    for(j=1; j<=n; j++){
        if(i==1 || i==n || j==1 || j==n)
            s.o.p("*");
        else
            s.o.p(" ");
    }
    s.o.println();
}
```

(51)

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

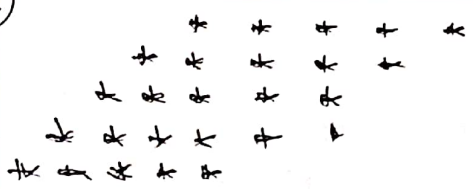
```
for(i=1; i<=n; i++){
    for(j=i; j<n; j++)
        s.o.p(" ");
    for(j=1; j<=n; j++){
        if(i==1 || j==n || i==j || j==1)
            s.o.p("*");
        else
            s.o.p(" ");
    }
    s.o.println();
}
```

(52)

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

```
for(i=1; i<=n; i++){
    for(j=1; j<=n; j++){
        if(i==1 || i==n || i==j || j==1 || j==n ||
           (i+j==n+1))
            s.o.p("*");
        else
            s.o.p(" ");
    }
    s.o.println();
}
```

(53)

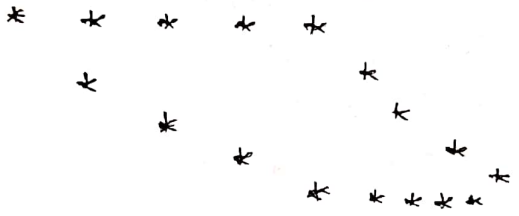


```
for(i=1; i<=n; i++) {
    for(j=i; j<n; j++)
        s.o.p(" ");
    for(j=1; j<=n; j++)
        s.o.p("*");
    s.o.pln();
}
```

(54)

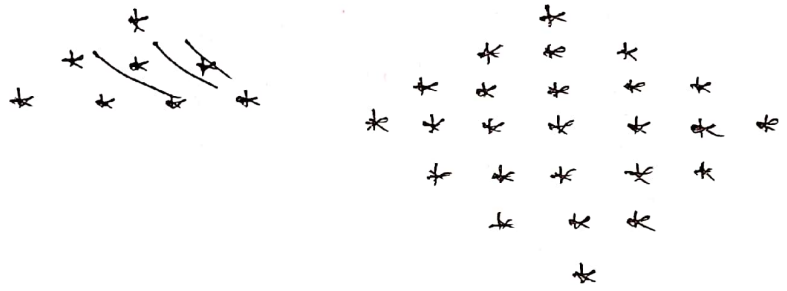
```
for(i=1; i<=n; i++) {
    for(j=i; j>1; j--)
        s.o.p(" ");
    for(j=1; j<=n; j++)
        s.o.p("* ");
    s.o.pln();
}
```

(55)



```
for(i=1; i<=n; i++) {
    for(j=i; j>1; j--)
        s.o.p(" ");
    for(j=1; j<=n; j++) {
        if(i==1||j==1||i==n||j==n)
            s.o.p("*");
        s.o.pln();
    }
```

(56)



```
for(i=1; i<=n; i++) {
    for(j=i; j<n; j++)
        s.o.p(" ");
    for(j=1; j<=2*i-1; j++)
        s.o.p("*");
    s.o.pln();
}
for(i=n-1; i>=1; i--) {
    for(j=i; j<n; j++)
        s.o.p(" ");
    for(j=1; j<=2*i-1; j++)
        s.o.p("*");
    s.o.pln();
}
```

(56)

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

```
for (i=1; i<n; i++) {
    if (i<=n/2) {
        for (j=1; j<=i; j++)
            S.o.p.("*");
    }
    else {
        for (j=i; j<n; j++)
            S.o.p.("*");
    }
    S.o.println();
}
```

(57)

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

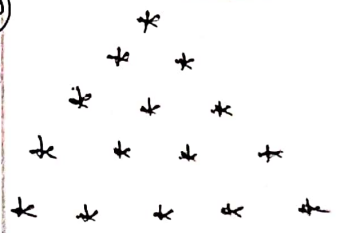
```
for (i=1; i<n; i++) {
    if (i<=n/2) {
        for (j=i; j<=n/2; j++)
            S.o.p.(" ");
        for (j=1; j<=i; j++)
            S.o.p.("*");
    }
    else {
        for (j=i; j>n/2; j--)
            S.o.p.(" ");
        for (j=i; j<n; j++)
            S.o.p.("*");
        S.o.println();
    }
    S.o.println();
}
```

(58)

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

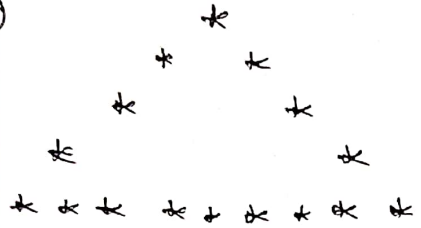
```
for (i=1; i<n; i++) {
    if (i<=n/2) {
        for (j=i; j<=n/2; j++)
            S.o.p.("*");
    }
    else {
        for (j=i; j<=n; j++)
            S.o.p.("*");
    }
    S.o.println();
}
```

(60)



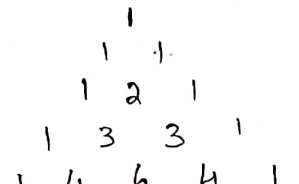
```
for (i=1; i<=n; i++){
    for (j=i; j<n; j++)
        S.o.P(" ");
    for (j=1; j<=2*i-1; j++)
        S.o.P("*");
    S.o.println();
}
```

(61)



```
for (i=1; i<=n; i++){
    for (j=i; j<n; j++)
        S.o.P(" ");
    for (j=1; j<=2*i-1; j++){
        if (j==1 || j==2*i-1 || i==n)
            S.o.P("*");
        else
            S.o.P(" ");
    S.o.println();
}
```

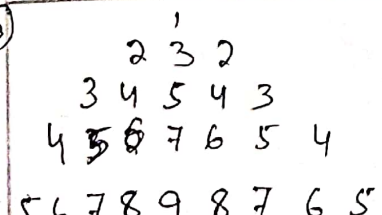
(62)



```
for (i=0; i<n; i++){
    for (j=1; j<n-i; j++)
        S.o.P(" ");
    for (j=0; j<=i; j++){
        if (i==0 || j==0)
            c=1;
        else
            c=c*(i-j+1)/j;
        S.o.P(c+" ");
    }
    S.o.println();
}
```

14.11.23

(63)



```
for(i=1; i<=n-1; i++){
    for(j=1; j<=n-i; j++){
        S.o.P(" ");
    k=i;
    for (j=1; j<=i; j++)
        S.o.P(k++);
    p=k-2;
    for (j=i; j>1; j--)
        S.o.P(p--);
    S.o.println();
}
```

(4)

```
    1
  1 2 1
 1 2 3 2 1
1 2 3 4 3 2 1
1 2 3 4 5 4 3 2 1
```

```
for(i=1; i<=n; i++){
    for(j=i; j<n; j++)
        S.o.p(" ");
    for(j=1; j<=i; j++)
        S.o.p(j);
    for(j=i-1; j>=1; j--)
        S.o.p(j);
    S.o.println();
}
```

(5)

```
    1
  2 1 2
 3 2 1 2 3
4 3 2 1 2 3 4
5 4 3 2 1 2 3 4 5
```

```
for(i=1; i<=n; i++){
    for(j=i; j<n; j++)
        S.o.p(" ");
    for(j=i; j>=1; j--)
        S.o.p(j);
    for(j=2; j<=i; j++)
        S.o.p(j);
    S.o.println();
}
```

UNCONDITIONAL STATEMENT

→ These are the statements which moves from one point to another point of the program without evaluating conditions.

→ Different types of unconditional statements are

(1) break
(2) continue
(3) return.

Break

→ A break in a switch will make us to come out of the switch statement.

→ Similarly a break in a loop will make us to exit from the loop and proceed with a next statement outside the loop.

Continue

→ A continue statement will skip the remaining statements after the continue keyword in a loop and makes the control to go to the beginning of the loop for the next iteration.

→ A continue statement is also treated as a bi-passer.

Ex:

```
for (i=1; i<=10; i++){
    if (i==5)
        continue;
    else
        s.o.p.(i);
}
```

o/p → 1 2 3 4 6 7 8 9 10

```
for (i=1; i<=10; i++){
    if (i==5)
        break;
    else
        s.o.p(i);
}
```

o/p → 1 2 3 4

Return

→ Return statement is used to return some value.

→ In java, return statement is used for returning a value when the execution of the block is completed. The return statement inside a loop will cause the loop to break and further statements will be ignored by the compiler.

→ Every method must return some value. Based on the declared datatype of the method, the method will return value.

→ Ex:

```
int add( ){
    int a=5;
    int b=2;
    int sum=a+b;
    return sum;
}
```

→ Return is the last statement of a method.

NOTE

CLASS is a logical representation of data

Passing Object as Argument inside a method

→ In this method, we have to pass the object as a method parameter to call a method.

→ Ex:

```
class Student
{
    int rollno=1234;
    String name="sachin";
    String branch="CSE";
    int semester=4;
    float marks=456.0f;
}
```

```
class Objectdemo {
    Public static void main (String [] a
    { Objectdemo o=new Objectdemo ();
      Student s=new Student ();
      o.display(s); // calling Method
    }
    Void display (Student s1){ // called Method
      S.o.pln (s1.rollno);
      S.o.pln (s1.name);
      S.o.pln (s1.branch);
      S.o.pln (s1.semester);
      S.o.pln (s1.marks);
    }
}
```

Method Returning an Object

→ In this process the returntype of the method is of class & method will return object of the class.

→ Ex:

```
class Student {
    int rollno=1234;
    String name="sachin";
    String branch="CSE";
    int semester=4;
    float marks=456.0f;
}

class Objectdemo1{
    Public static void main (String [] args) {
        Student s=new student ();
        Objectdemo1 o1=new Objectdemo1 ();
        s=o1.display ();
        S.o.pln (s.rollno);
        S.o.pln (s.name);
        S.o.pln (s.branch);
        S.o.pln (s.semester);
        S.o.pln (s.marks);
    }
    Student display () {
        Student s1=new student ();
        return s1;
    }
}
```

METHOD CALLING

→ Method calling is required for performing some certain operation based on method.

→ For static method, we can call the method by

```
Classname.methodname();
```

→ For instance method, we can call the method by creating the object for the class.

```
Classname objectname = new classname();
objectname.methodname();
```

→ Method calling can be done by 4 methods/types

(1) Method with arguments with return value.
(2) Method with arguments without return value
(3) Method without arguments with return value.
(4) Method without arguments without return value.

Method with arguments with return value

→ In this type of method calling, we have to pass arguments to the methods & the method have a return value.

→ Ex:

```
class Triangleareа {
    public static void main (String [] args) {
    Scanner sc = new Scanner (System.in);
    float a, b, c, area;
    a = sc.nextFloat();
    b = sc.nextFloat();
    c = sc.nextFloat();
    area = areaOfTriangle (a, b, c); // calling method
                                     // arguments
    S.o.p. ("Area of Triangle = " + area);
    }
Static float areaOfTriangle (float a, float b, float c) {// called method
                                  // arguments
        float s, area;
        s = (a+b+c)/2;

        area = (float) Math.sqrt (s * (s-a) * (s-b) * (s-c));
        return area;
        }
    }
```

→ Here a, b, c in main method and a, b, c in areaOfTriangle method are different. They are local variable to the concerned method.

Method with argument without return value

→ In this type of method calling, we have to pass some value as as arguments but method don't return any value.

→ Ex:

```
class Greater{
    Public static void main(String [] args){
        Scanner sc= new Scanner(System.in);
    int a=sc.nextInt();
    int b=sc.nextInt();
    int c=sc.nextInt();
     Greater g=new Greater();
         g.greater(a,b,c); // calling method
     }
     void greater(int a, int b, int c)// called Method.
      {
        if (a>b){
              if (a>c)
                  S.O.P.(a+" is greatest");
              else
                  S.O.P.(c+" is greatest");
              }
          else{
              if (b>c)
                  S.O.P(b+" is largest");
              else
                  S.O.P(c+" is largest");
              }
        }
    }
```

Method without arguments with return values

In this type of method calling, we don't pass any values as arguments but method return some value.

Ex:

```
class prime{
        Public static void main(String args[]){
        boolean flag = isPrime();
        if(!flag)
            S.O.P.("Prime number");
        else
            S.O.P("non-prime number");
        }
        static boolean isPrime(){
            Scanner sc=new Scanner(System.in);
            int n = sc.nextInt();
            boolean flag=false;
```

```
for (i=2; i<=n/2; i++) {
    if (n%i==0) {
        flag=true;
        break;
    }
}
return flag;
}
}
```

Method Without argument without return type value

→ In this method we don't pass any value as argument and method also don't return any value.

→ It is just like as writing methods is main method.

→ It is the least preferred method calling.

RECURSION

→ Recursion is a process in which a method calls itself repeatedly.

→ Internally, recursion uses stack.

→ For recursion we must need a stopping condition.

→ Whenever a method calls, the address of the method is stored in the stack & according to the address, the o/p returns to the calling method (LIFO).

→ If recursion don't have a stopping condition then every time the method calls itself, the address stores in stack and at a certain point of time the stack will full and we will get stack overflow error. That's why stopping condition is required for recursion.

① Write a java program to find the product of 2 numbers using recursion.

```
p.s.v.main() {
a=sc.nextInt(); b=sc.nextInt();
prod(a,b); //(12)
}
static int prod(int x, int y) {
    if (x==1)
        return y;
    else
        return (y+prod(x-1,y));
}
```

Let a=3, b=4

prod(3,4);

x=3, y=4

prod(3,4)

3==1 ✗

return(4+prod(2,4)); → (11)

prod(2,4)

2==1 ✗

return 4+prod(1,4)

4+4=(8)

prod(1,4)

1==1 ✓

return (4)