Generation of languages

1) First generation language: Machine language/ low level language is the first generation of language.

Advantages of Machine language.

1) Machine language makes fast and efficient use of the computer.
2) It requires no translator to translate the code. It is directly understood by the computer.

Disadvantages of Machine language.

1) It's very time consuming. (It takes months sometimes years to make a small program)
2) It's very hard to remember all ~~ASCII~~ ASCII code.
3) Error detection and correction were very very hard / difficult.
4) It's machine dependent.

2) Second generation language: Assembly language/ Middle level language. is the second generation of language

Advantages of Assembly language

1) Programs written in assembly language can be faster than those written in higher-level languages because they are closer to the machine code that the computer executes and do not requires the additional steps.

2) Assembler can be smaller in size than programs written in higher-level language

3) Accessibility is more than any machine language

4) Learning can be more easy than low level language as it uses Mnemonics.

Disadvantages of Assembly language

1) It also consumes time.

2) It's very hard to remember all Mnemonic codes.

3) It needs a translater called Assembler (an assembler is a translator which converts assembly language into machine language)

4) It's also machine dependent.

eg: of Mnemonic's : Add, load, Subb, Inc, Dec, Div, Mul etc.

3) Third generation of language: High level language

Advantages of high level language

1) It is simple to write.
2) Easy to read and maintain
3) High-level language offers built in function library which boost productivity.
4) It reduces error
5) It has enhanced security.

Disadvantages of high level language

1) It is generally slower than lower level lang
2) It has less control on code optimization for a specific hardware or system
3) higher memory usage than low level language.

e.g of high level language:

Cobol → Algol 60 → C - P - L → B → C

language → B language → C - language (1972)

C++ → Java
1979

What is Java?

Java is a general purpose object oriented programming language through which we can communicate with our computer. It is developed by Sun microsystem of USA in year 1991. Originally Java was called Oak by James Gosling, one of the inventor of Java. Java was designed for the development of software for consumers electron devices like TVs, vehicles, clocks, tosters and such other electronic machines. The goal has a strong impact on the development team to make the language simple and highly reliable. The most striking feature of this language is that it is a platform neutral language. Java is the first programming language that is not tied to any particular hardware or operating system. Programs designed in Java can be excuted anywhere on any system. We can call Java as a revolutionary technology because it has bought a fundamental shift on how we develop and use programs. Nothing like this has happened to the software industry before.

Features of Java

The inventor of Java wanted to design a language which could offer solution to some of the problems encountered in modern programming. They wanted the language to be not only reliable, portable and distributed but also simple, compact and interactive. Some

important features of Java are as following:

1. Compiled and interpreted: Java is the only language which is compiled as well as interpreted.
2. Java is a platform independent and portable language.
3. Java is a pure object oriented language.
4. It is robust and secure.
5. It is also distributed language.
6. Java is a familiar, simply and small language.
7. It's performance is very high.
8. It is dynamic & extensible.
9. We can create desktop client applications in Java.
10. It also supports XML processing and web services.

Java Environment Variable:

Java environment variable includes a large number of development tools and 100's of classes and methods. The development tools are part of the system known Java Development Kit (JDK) and the classes and methods are part of the Java

library also known as the application program interface (API).

Java Development Kit (JDK):
Java development kit comes with a collection of tools that are used for developing and running Java programs. They include:

1) Applet Viewer (for viewing Java applet)
2) Javac (Java compiler)
3) Java (Java interpreter)
4) JavaP (Java disc assembler)
5) JavaH (for C header files)
6) Java doc (for creating HTML documents)
7) JDB (Java debugger)

How Java program is executed

1) Writing Java program.
Java program start with writing the source code (main file) in a .java file using a text editor like, Notepad, C++, VS code etc.

2) Compilation
The Java compiler (Javac) compiles the .java file into bytecode, generating a .class file.

that bytecode is platform-independent

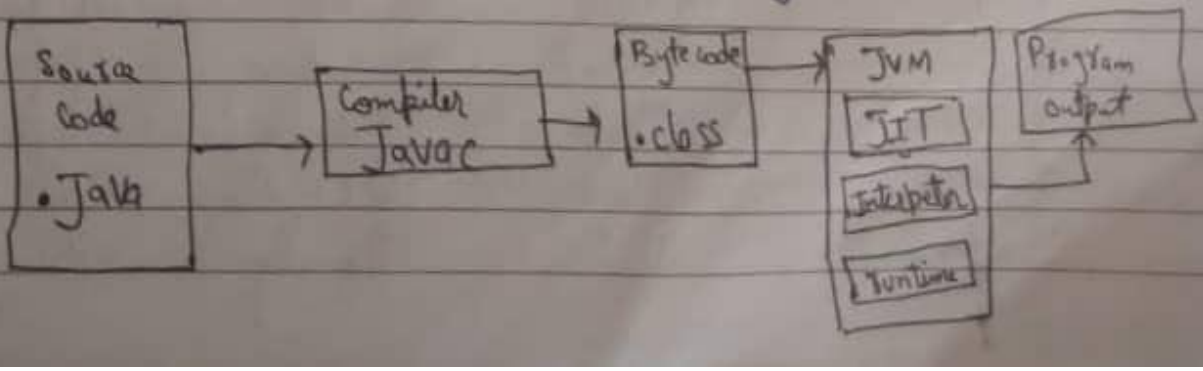
3. Java Bytecode
The compiled bytecode (.class file) is an intermediate representation of your program. It can run on any platform with a Java virtual machine (JVM).

4. Java virtual Machine (JVM)
The JVM is a crucial component that interprets bytecode and executes it on the underlying hardware.

5. Class loader.
The class loader loads the .class files into memory. It finds and loads the bytecode for all classes needed to run the program.

6. Bytecode Verification
The JVM verifies the bytecode to ensure it doesn't violate Java's security constraints. This step prevents malicious code for executing.

7. JIT Compilation and Execution
The Just-In-Time (JIT) compiler translates bytecode into native machine code for faster execution. The JVM then executes this native code on the CPU, running the



Java - Tokens:

In Java, Tokens are the smallest elements of a program that is meaningful to the compiler. They are also known as the fundamental building blocks of the program. Tokens can be classified as follows:

1. Keywords.
2. Identifiers
3. Constants
4. Special symbols
5. operators.
6. Comments.
7. Separators

1. Keywords: Keywords are pre-defined or reserved words in a programming language. there meaning is already defined to our compiler/interpreter and we cannot change the meaning of these keywords. Keyword's cannot be used as variable name, in Java. Java support the following keywords.

1) abstract	10) Int	19) try	28) goto
2) break	11) module	20) Volatile	29) import
3) catch	12) open	21) assert	30) interface
4) Const	13) private	22) byte	31) native
5) do	14) Public	23) char	32) opens
6) enum	15) Short	24) Continue	33) protected
7) final	16) Super	25) double	34) requires
8) for	17) This	26) exports	35) static
9) implements	18) To	27) finally	36) Switch

37) Throw 47) extends 55) strictfp
38) transient 47) float 56) Synchronized
39) Uses 48) if 57) throws
40) while 49) Instanceof 58) transitive
41) boolean 50) long 59) Void
42) Case 51) new 60) with
43) class 52) Package
44) default 53) Provides
45) else 54) return

2. Identifiers: Identifiers are used as the general terminology for naming of variables, functions and arrays. These are user-defined names consisting of an arbitrarily long sequence of letters and digits with either a letter or the underscore (_) as a first character.

Rules to declare an Identifiers:

1) Identifiers cannot begin with space or have space
2) Should not begin with a digits.
3) Should be alphanumeric character.
4) We cannot use keywords as ~~void~~ Identifiers.

3) Constants / literals: Any constant value which can be assigned to the variable is called literals / constant. They are synthetic representation of boolean, numeric, character, or string data. It is a medium of expressing particular values in the programs.

2) Unary operator: These are the type of operators that needs only one operand to perform any operation like, increment, decrement, negation etc.

3) Assignment operator: These operators are used to assign values to a variable. The left side operand of the assignment operator is a value. The value on the right side must be of the same datatype of the operand on the left side. Otherwise, the compiler will raise an error.

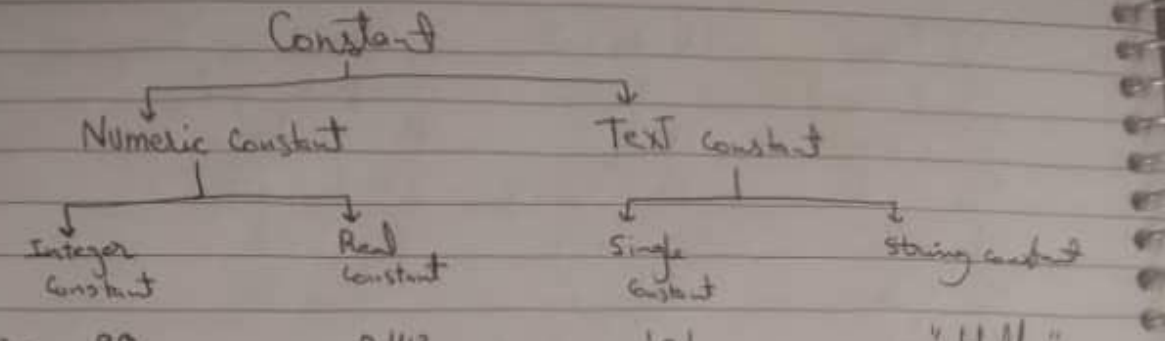
Assignment operator

- Simple: a = 10
- Compound: ~~a = a+10~~ a+=10

4) logical operator: These are used to perform "And", "Or" and "Not" operations

And	&&
Or	||
Not	!

6 Comments: These are the part of the program which are ignored by the compiler while compiling the program. They are useful as they can be used to describe the operation or methods in the program. These are of Two types



e.g;	98	0.413	'a'	"Hello"
	4721	8.55	'z'	"October"
	+22418	+24.75	'S'	"Swift"
	-3174	-0.1159	'u'	"Computer"
	41445	41.331	'1'	"Number"

4. Special Symbols: The following special symbols are used in Java having some special meaning and cannot be used for some other purpose
[ ], ( ), { }, ; , *, =

- Brackets [ ]: Opening and closing brackets are used as array element reference.
- Parenthesis ( ): These special symbols are used to indicate function calls and function parameters.
- Braces { }: These opening and ending curly braces marks the start and end of a block of code containing more than one executable statement.
- Comma (,): It is used to seperate more than one statements like for separating parameters in function calls.

- Semicolon ; It is an operator that essentially invokes something called an initialization list.
- Asterick (*): It is used to create pointer variable.
- Assignment operator (=): It is used to assign values.

5. Operators: Java provides many types of operators which can be used according to the need. Some of the types are:

1) Arithmetic operators: These involves the mathematical operators that can be used to perform various simple or advanced arithmetic operations on the primitive data types referred to as the operands.

Operators	Output
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Module / return of remainder

- Single line comment : //
  used for commenting a single line.
- Multi line comment : /* */
  used for commenting multiple lines.

7. Seperators: These are used to seperate different parts of codes. It tells the compiler about completion of a statement in the program. The most commonly and frequently used seperator in Java is Semicolon(;)

Types of Errors In Java

1) Run-time error: Run time error or we can say, are detected during the execution of the program. Sometimes these are discovered when the user enters an invalid data or data which is not relevant.

2) Syntax Error: Spelling or grammatical mistakes are syntax error, for e.g using an uninitialized variable, using an undefined variable, etc, missing a semicolon, etc.

3) logical Error: A logic error is when your program compiles and executes, but does the wrong thing or returns an incorrect result or no output when it should be returning an output. These errors are detected detected detected neither by the compiler nor by JVM. The Java System has no idea what your program is supposed to do, so, it provides no additional

Simple Structure of JAVA:

// Documentation part
link part
class class-name
{
public static void main (String ↓ args [])
command line argument
{
Declaration:
Initialization
Processing
}
}

(5) letter of the above word

// prog. to print a simple text message.

```
Class hello
{
  Public Static void main
          (String args[])
  { System.out.print (welcome to JAVA
        programming)
    }
}
```

S of System should be capital

→ Int allocates 4 byles of memo in JAVA

→ The work of nextInt is to take value from the keyboard

// Prog. to add 2 no's

```
Class Addition
{
    PSV main (String args[])
{
    Int x, y, Z;
        x = 10;
        y = 20;
        Z = x + y;
    S.O.P("the sum of 2 no's
            its = " + Z);
}
}
```


User Interactive // Prog to add 2 no's

```
Import Java util Scanner class
Addition
{
    PSV main (String args [])
    {
        Int x, y, Z;
```

Class → Scanner input = new Scanner (System in

object name

```
S.O.P (enter 1st no = ");
    x = input nextInt ();
S.O.P (enter 2nd no = ");
    y = input next Int ();
        Z = x + y;
S.O.P ("The sum of 2 no's = "
                + Z
```

classmate Date Page 08

// Prog to prepare the marks card of a student

Import java.util.Scanner
class Markscard

{ p s v main (String args[])

{ Int Eng, Math, urdu, sci
SSt

Total marks, Per;

Scanner input = new Scanner (System.in)

S.o.P("enter the mark of 5 sub=");

Eng = input.nextInt();

math = // //();

urdu = // //();

classmate Date Page 09

Sci = // //();
Sst = // //();

Total mark = eng + math + urdu + sci + sst

Per total marks * 100/500;

S.o.P("The total marks of a Stu = " + total marks)

("The per of a student = " + per)

}

}

If

// Preparing Mcard using If Condition

If (Per) =50) - Pass

// (// < 50) - Fail

After printing the value of %age

If (Per >=50)

So.p (" you have passed "); ×

If (Per >=50)

So.o.p (you have failed ")

}

}

If - else

If (Per >=50)

S.o.p (" you have passed ");

eles

S.o.p (you have failed ");

Nested If -eles

If (Per <= 33)

S.o.p (the student is fail ");

else

if (Per >33 && Per <=50)

S.o.p (student has C- grade ");

else

if Per >50 && Per <=80 ");

S.o.p (student has B-grade "

else

if (Per >80 && Per <=90 "

S.o p (Student has grade
else
S.o p (Students has Distiction
}
}
}

Loops

Series Generation

// Prog to gen a simple Series
class Series
{
p s v main (String args [])
{
int i;
i = 1;
while (i <= 10)
{
Sop (i)
i++
}
}
}

User Interactive

```
import java.util.Scanner;
class Series
{
    p s v main (String args [])
    {
        int start, limit;
        Scanner input = new Scanner
                        (System.in);
        S.O.P (enter the value for
               start and limit)
        start = input.nextInt();
        limit = ll ll ();
        while start <= limit
        {
            S.O.P (start);
            start++;
```

Even Series

```
{
    Int i;
    i = 1;
while (i<=10)
{
  if (i%2 == 0)
  S.op (i);
    i++
};
};
```

OR

```
{
    Int i;
      i = 2
    while (i<=20)
{
      S.op (i);
      i+2;
```

Odd Series (User Int;

Import Java utile Scanner class

Series

```
{ p s v main (String args [])
  {
      Int Start limit
Scanner input = new Scanner (System
  S.O.P " enter the value for Star
          and limit
Start = input next Int (1;
  while (Start <= limit)
{
  if (Start % 2 == 1)
```

```
System out print (staut)
    staut ++;
}
}
```

Table generation

```
class table
{ ps v main (string aug (1)
{   gnt i;
    1=1;
  while (i<= 10)
{
  System out print ("2
x "+i +cc= "+
  2*1);
```



```
i ++;
}
}
}
```

table gen (user Int.)

Import Java.util.Scanner class table

{

ps v main (String args[])

{

Int start, limit, Tab;

Scanner input = new Scanner (System.in);

S.o.p("enter the value for start limit and table

start = input.next Int ();

limit = " " (1;

tab = " " (1;

while (start <= limit

System.out.print ("Tab + "x" + start + "=" + tab * start);

start ++;

}

}

Object Oriented Programming

// Prog to add 2 no's

class Addition

{ int x; } - data member

int y, z;

void getdata

{ x = 10;

y = 20;

}

void Showadd ()

{ z = x + y;

S.O.P ("the sum is =" +

}

}

class addmember

{ PSVM (String args[]

{ addition a1 = new addition

// a2 = // // ();

a1 getdata ();

a2. // //

a1 Showadd ();

a2 // //

}

}

```
Class comparison
{   int x, y;
    void setdata (int a, int b)
    {   x = a;
        y = b;
    }
    void compare()
    {   if (x > y)
        S.o.p (x + " is greater")
        if (y > x)
        S.o.p (y + " " + "\n");
    }
else
```



```
S.o.p ("Both no's are same");
}
}
Class check
PSV main String args[])
{
    Comparsion c1 = new Compars
    // // c2 = // // c1;
    c1 setdata (10, 20);
    c2 // (100, 200);
    c1 compare ();
    c2 // c1;
}
}
```

Swapping of no's

```
Import Java util Scanner class
Swap
{
    P.S V main (String args[])
        Int x, y, temp;
    Scanner input = new Scanner
                    (System in)
S.o.p (enter Ist no = "");
        x = input.next Int ();
S.o.p (enter 2nd no = "");
    y = input next Int ();
      temp = x;
       x = y;
```



```
      y = temp
S.o.p (value after
        Swapping = "");
S.o.p ("x = " + x);
    // ("y = " + y);
    }
}
```

// Prog to Swap 2 no's (= OOPs)

```
import java.util.Scanner class Swapping
{
    int x, y, temp
    Scanner sc = new Scanner (System.in);
    voied getdata
{
    S.o.p ( Enter 1st no=");
    x = scan.nextInt();
S.o.p (enter 2nd no=");
    y = Scan.nextInt();
}
```



```
void showdata()
{
    S.o.p ("x =" + x);
    // ("y =" +y);
}
void swap ()
{
    temp = x;
    x = y;
    y = temp;
}
}
```

```
Class Interchange
{
  PS V main String args(L)
  {
    Swapping s1 = new Swapping 1);
    s1.getdata (1;
    S.O.P ("Data before Swapping")
    s1 Show data ();
    S.O.P ("Data after Swapping
      s1 Swap (1);
      s1 Showdata (1);
  }
}
```

```
// Prog to check whether the no is even or odd
Import Java.util Scanner
Class check
{
  Int x;
  Scanner input = new Scanner
                  (System.in)
  S.op ("enter any no =");
  x = input next Int( );
    if (x % 2 == 0)
      S.O.P ( x + "is even")
  if (x % 2 == 1)
      S.O.P ( x + "is odd"
}
```

Even/odd (oops).

```
Import Java util Scanner class
Number
{
    Int x;
Scanner input = new Scanner
                (System in)
    Void getdata()
{
    S.o.p enter any no");
      x = input next Int();
}
    Void Showdata()
{
```



```
    if (x % 2 == 0)
S.o.p (x + "is even");
  else
      S.o.p (x + "is odd");
}
    Class check
{
  P S V main (String args[])
  {
    Number = new number()
      N1 getdata();
      N1 Showdata();
  }
}
```

// Prog to get the area of O

```
import java.util.Scanner.area
{
  P.S.V main (String args [])
  {
    Single radius area.
    Scanner input = new Scanner
                    (System.in)
    S.O.p ( enter the radius
    radius = input.next Int ();
    area = 3.14 * radius * radius
    S.O.P [The area of a circle
          = + area
  }
}
```


Area of a O(oops)

```
Imports Java.util.Scanner
  class area
{
  Single radius, area
  Scanner input = new Scanner
                  (System.in)
  void get data
{
  S.O.p (enter the radius")
  radius = input.next int ();
}
  void showdata
{
```

area = 3.14 * radius * radius
S.O.P ("the area of a Circle")
}

Class Circle

{
P.S.V main (String args[])

{
Area a1 = new are ();
a1 get data ();
a1 Show data ();
}
}

Cube of a no (OOPS)

Import Java.util.Scanner class
Cube

Int C, Num
Scanner input = new Scanner (System.in)

Void getdata ()

{
S.O.P ("enter the value for C = ");
C = input.next int ();
}

Void Showdata

{
Num = C * C * C;
S.O.P ("the cube of a no is = " + num);

```
class Check
{
    P.S.V main String arg[])
    {
        Cube C1 = new cube();
        C1 get dāla();
        C1 showdāla();
    }
}
```


How to use object as a function argument

```
class Swapping
{
    Int x;
    Int y;
    void Set dāla(int a, int b)
    {
        x = a;
        y = b;
    }
    void Showdata()
    {
        S.O.P("x = +x");
    }
    Void Swap Swapping S2)
```

```
{
    int temp;
    temp = x.x;
     x = s2.x;
    s2.x = temp
     temp = y;
    y = s2.y;
     s2.y = tem;
}
}
class Interchange
{
    p s v main String args[]
```



```
{
    Swapping s1 = new Swapping ();
Swapping s2 = new Swapping ();
    s1.setdata (10,100);
    s2 "  (20,200);
    S.o.p(" value before swapping
    s1.showdata ();
    s2 "  "
    s1.swap (s2);
    S.o.p ("Data after swapping
    s1.showdata ();
    s2. "  "
}
}
}
```

How to return an object

```
import java.util.Scanner
class add data
{
    int x;
    // y;
Scanner input = new Scanner
                (System.in);
void Set data
{
    S.O.P (enter any 2 no's
    x = input.next (1);
    y =  //    //    //
}
void Show data ( )
```



```
{
    S.O.P ("x = " + x);
    // ("y = " + y);
}
add data Sum Obj (add data a
    add data temp = new add data
    temp.x = x + a2.x;
    // .y = y + a2.y;
    return (temp);
}
}
```

```
class addition
{
    p s v main (String args[])
    {
        odd dala a1 = new add dala
          //  a2 =  //   //
          //  a3 =  //   //

        a1 get dala ();
        a2. //   //
        a3 = a1 . Sum00 (a2);

        S. o.p . The addition of 2
                 objects is = ;
        a3 Shorddala ( );
    }
}
```

Function: A set of instruction there are 4 types of function

1 function = no augument and no return
2 // // // but return
3 // // // augoment but no return
4 // // // and & return

↝ function = no augument and no return

```
Class Addition
{
    PS v main (String args[])
    {
        get add ()
        S.O.P (" I am in main program );

        void get add ()
        {
```

```
Int x, y, Z;
    x = 10;
    y = 20;
    Z = x + y;
S.O.p (The Sum is = "+Z);
}
}
```



```
// Prog to compare 2 no's
Class comparsion
{
    Ps v main (String args [])
    {
        compaee ();
    }
    void compare ()
    {
        Int x, y;
            x = 10;
            y = 20;
        if (x > y)
        S.O.p (x + " + is greater");
        else
```

```
S.O.P (y + " is greater ");
}
}
```

function = no argument but return

```
Import java.util.Scanner
class Addition
{
    P.S.V main (String args[])
    {
        int Z;
        Z = getadd ();
        S.O.P (the Sum is = +Z);
    }
```



```
int compare
{
    int a, b, c;
    a = 10;
    b = 20;
    c = a + b;
    return (c);
}
}
```

22/08/2022.

Prog to Compare 2 no's

```
import java.util.Scanner;
class comparision
{
    p s v main (String args[])
    {
        int z;
        z = get comparision ();
        s.o.p ("The greater no is" + z
    }
    int compare ()
    {
        int x, y;
        Scanner input = new Scanner
        System
        s.o.p (enter any 2 no's );
        x = input.nextInt ();
        y = " "
        if (x > y)
        return (x);
        else
        return (y);
    }
}
```

To get factorial of a no.

```
class factorial
import java.util.Scanner
{
  p s v main (String args [])
  {
    int fact
    fact = getfact (1);
    S.o.p ("The factorial of a no. is
            = + fact)
  }
    int num, f = 1;
Scanner input = new Scanner
                (System.in)
S.o.p ( enter any no for
        factorial = ")
num = input.nextint (1);
while (num > 1)
{
  f = f * num;
  , num--;
  return (f);
}
}
```

function c Argument but no return

```
import java.util.Scanner;
class Addition
{
    P S V main (String args[])
    {
        int x,y;

Scanner input = new Scanner(System.in);
S.o.p(enter any 2 no's = "");
        x = input.nextInt();
        y = " "
    add (x,y);
}
    void add (int x int y)
{
    Int z
    z = x+y
    S.o.p (The add is = "+z);
}
}
```


⇝ function c Argument and c return

```
Import Java.util.Scanner class
Addition
{
    PS V main String args ())
    {
        int x,y - Z;
Scanner input = new Scanner(syste
```