* Simple Java program :-

→ In this section, we can discuss how to execute a Java program and what are the requirements to execute a Java program.

→ For executing any Java program, you need to

- Install the JDK (Java Development Kit).
- Set path of the jdk/bin directory.
- Create the Java program.
- compile and run the Java program.

* Install the JDK:

JDK - Java Development Kit

→ It is a software development environment for Java applications and applets.

→ JDK includes

- Java compiler (Javac)
  It translates the source code to byte code.
- Java debugging tool (Jdb)
  It is used to run Java program
- Java Runtime Environment (JRE)
  It provides an environment to run any Java program at any platform.
- Java Archiving tool (jar)
  It is used to distribute the Java Appln through network with .jar extension.

→ JDK is available for free at www.oracle.com under Java SDKs and tools → Java SE.

→ Download the latest JDK and install it.

→ on Windows, the JDK will be installed by default directory i.e " C:\ program files \ Java \ jdk 1.8.xx ".

* Set path of the JDK/bin directory:

The path is required to be set for using tools such as Javac, Java etc.

For setting the permanent path of JDK, you need to follow these steps:

- Goto My computer properties → advanced tab → environment variables → new tab of user variable → Write 'path' in variable name → Write 'path of bin folder' in variable value → OK → OK → OK.

→ The path of bin folder is look like

"C:\program Files\Java\jdk1.x.x\bin".

→ To verify that JDK is properly installed, open the command prompt type "Javac" and press "Enter".

* Create the Java program:

→ To create Java program open any editor such as notepad.

→ Type the source code

Ex:-

```
class Sample
{
  public static void main(String args[])
  {
    System.out.println("Hello Java");
  }
}
```

→ Save this file as "Sample.Java".

* Compile and run the Java program:

→ To compile 'Javac' command is used

for ex: Javac Sample.Java
→ We get "Sample.class" file

→ To execute 'Java' command is used

for ex: Java Sample
→ We get o/p 'Hello Java'.

In the above Java program, let's see what is the meaning of class, public, static, void, main, String[], System.out.println().

- class is a keyword is used to declare a class in Java.
- public keyword is an access modifier which represents visibility, it means it is visible to all.
- static is a keyword, if we declare any method as static, it is known as static method. The core advantage of static method is that there is no need to create object to invoke the static method.
- void is the return type of the method, it means it doesn't return any value.
- main is a method, it represents startup of the program. The main method is executed by the JVM.
- String[] args is used for command line arguments.
- System.out.println() is used print statement.
  - System: It is a class, which belongs to Java.lang package.
  - out: It is an output stream object, which is a member of System class.
  - println(): It is a method supported by the output stream object "out". It is used to display any kind of output on the screen.

→ At compile time, Java file is compiled by Java compiler and converts the Java code into bytecode (class file).

→ At runtime, class file is converted into machine understandible Instructions.

* Java comments :-

→ The comments are statements that are not executed by the compiler and interpreter.

→ The comments can be used to provide information, explanation and hide program code.

There are three types of comments in Java.

1. Single Line Comment
2. Multi Line Comment
3. Documenation Comment

1. Single Line comment:

This is used to comment only one line.

Syntax: // This is single line comment

2. Multi Line comment:

This is used to comment multiple lines of code.

Syntax:
```
/*
This
is
multi line
comment
*/
```

3. Document comment:

This is used to create documentation API. To create documentation API, you need to use "javadoc tool".

Syntax:
```
/**
This
is
document
comment
*/
```

→ The javadoc tool creates HTML files for your program with explanation.

* Data types in Java :-

→ Data types represents the different values to be stored in the variable.

→ Java defines eight data types, those are

- byte
- short
- int
- long

} Integer group

- float
- double

} Floating-point group

- char ; character group
- Boolean ; Boolean group

Datatype	Default Size	Range
byte	1 byte	-128 to 127
short	2 byte	-32,768 to 32,767
int	4 byte	-2147483648 to 2147483647
long	8 byte	-9,223,372,036,854,775,808 to 9,2..807
float	4 byte	$-1.7 * e^{38}$ to $1.7 * e^{38}$
double	8 byte	$-3.4 * e^{308}$ to $3.4 * e^{308}$
char	2 byte	0 to 65,536
Boolean	1 bit	0 or 1.

* variables :-

Variable is a name of memory location, in that we can able to store the value for the particular program.

There are three types of variables

→ Local variable

→ Instance variable

→ Static variable

Ex: `int x = 10;` // where x is variable name.

→ Local variable:

A variable which is declared inside the method is called local variable.

→ Instance variable:

A variable which is declared inside the class but outside the method, is called instance variable. It is not declared as static.

→ static variable:

A variable that is declared as static is called static variable. It cannot be local.

Example:-

```
class  Vardemo
       {
         int x = 50; // instance variable
         static int m= 10; // static variable
         void mains()
          {
            int n=9; // local variable
          }
       }
```

* constant:-

→ There are several values in the real world which will never change, those are called as constants.

eg:- PI (π) values is 3.142 and a day will always have 24 hrs.

→ A constant in java is used to map or assign an exact and unchanging value to a variable.

→ Java does not directly support constants, However, a static final variable is effectively a constant.

Example: public static final int MAX_VALUE =25;

* The scope and Lifetime of variables :-

→ Each variable in modern programming language has:
- a name
- an address
- a type

→ In addition to above properties, each variable also has:
- a scope
- a Lifetime

• scope:

→ The scope of a variable is the locations/places/range in a program where the variable is accessible/visible.

→ We can declare variables within any block.

→ Block is begun with an opening curly brace and ended by a closing curly brace.

→ one block equal to one new scope in Java.

→ A scope determines what variables are visiable to other parts of your program and also determines the lifetime of those objects.

• Lifetime:

→ The Lifetime of a variable is the location (i.e place) where the variable exists.

→ The Lifetime refers to the amount of time a variable exists.

→ variables are created when their scope is entered, and destroyed when their scope is left. This means that a variable declared with in a method will not hold their values outside the method.

→ variables are created and destroyed while the program is running.

* operators :-

An operator is a symbol that is used to perform operations. There are many types of operators in java such as

- Arithmetic operators

  +, -, *, /, %, ++, --

- Relational operators

  ==, !=, >, <, >=, <=

- Logical operators

  &&, ||, !

- Bitwise operators

  & - Bitwise AND
  | - Bitwise OR
  ^ - Bitwise exclusive OR

Ex:

a	b	a & b	a | b	a ^ b
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

<< - Left shift. Ex: a = 0001000, b = 2
>> - Right shift.

a << b 0100000
a >> b 0000010

- Assignment operators

  =, +=, -=, *=, /=, %=

- conditional operator

  exp ? value1 : value2

* operator precedence & hierarchy :-

Operators	precedence
postfix	expr++, expr--
prefix	++expr, --expr
Multiplicative	*, /, %
additive	+, -
shift	<<, >>
relational	<, >, <=, >=
equality	==, !=
bitwise AND	&

bitwise exclusive OR	^
bitwise OR	|
logical AND	&&
logical OR	||
conditional	? :
Assignment	=, +=, -=, *=, /=, %=

* Expression:-

An expression is a construct made up of variables, operators and method invocations, which are constructed according to the syntax of the language, that evalutes to a single value.

Examples:

```
int marks =25;
int Extmarks= 75;
int Total =0;
Total = marks + Extmarks;
```

* Type conversion and casting:-

• Type conversion:-

It converts the one data type into another. If both are compatiable, then Java compiler will perform the type conversion automatically.

for example converting a int into float,
converting a float into double.

Ex:-

```
int i=100;
Long l= i;
float f= l;
```

o/p:
i value 100
l value 100
f value 100.0

→ Type conversion is done by Java compiler, but remember we can store a Large data type into the other.

• Type casting:-

When a user can convert the one higher data type into lower data type then it is called as the type casting.

If both are incompatiable types, we must type casting.

Example :-

```
double d = 10.04;
long L = (long)d;
int i = (int)L;
```

o/p: d value 10.04
L value 10
i value 10.

Note:- Type conversion is done by compiler and Type casting is done by user.

Example :-

```
class TypeConversion
{
public static void main(String[] args)
{
    int x = 1024;
    float y;
    y = x;
    System.out.println("y value is "+y);
}
}
```

o/p:- y value is 1024.0

Example :-

```
class TypeCasting
{
public static void main(String[] args)
{
    double d = 10.04;
    long L = (long)d;
    int i = (int)L;
    System.out.println("d value is "+d);
    System.out.println("L value is"+L);
    System.out.println("i value is"+i);
}
}
```

o/p:- d value is 10.04
L value is 10
i value is 10

* Enumerated Types:-

An enum type is a special datatype that contains fixed set of constants.

In Java programming, you define an enum type by using the 'enum' keyword.

Example:

```
public enum Day {
        Sunday, Monday, Tuesday, Wednesday, Thursday,
        Friday, Saturday
        }
```

you should use enum types any time you need to repre - sent a fixed set of constants.

program:-

```
public class EnumExample {
    enum Day {
    Sunday, Monday, Tuesday, Wednesday, Thursday, Friday,
     Saturday
     }
    public static void main (String[] args)
     {
        Day yesterday = Day.Thursday;
        Day today = Day.Friday;
        Day tomorrow = Day.Saturday;
     System.out.println ("Today is " + today);
     System.out.println ("Tomorrow will be" + tomorrow);
     System.out.println ("yesterday was" + yesterday);
     }
}
```

o/p: Today is Friday
Tomorrow will be Saturday
yesterday was Thursday

* Conditional statements:-

These are used to check the condition and execute the set of statements based on condition.

The Java supports following conditional statements

→ If-else statements

→ switch statement

→ If-else statement:-

If statement is used to test the condition. It checks boolean condition: true or false.

There are various types of if statement in Java.

- If statement
- If-else statement
- If-else-if ladder.

- **If statement:**

Syntax:-

```
if (condition)
{
  // code to be executed
}
```

- **If-else statement:**

Syntax:-

```
if (condition)
{
// code if condition is true
}
else
{
// code if cond'n is false
}
```

- **If-else-if ladder:**

Syntax:

```
if (condition1)
{
// code if condition1 is true
}
else if (condition2)
{
// code if condition2 is true
}
...
else
{
// code if all conditions are false
}
```

Example: Demonstrate if-else statements.

```
public class IfElseDemo
{
  public static void main (String [] args)
  {
    int marks = 76;
    char grade;
    if (marks >= 90)
    {
      grade = 'A';
    }
    else if (marks >= 80)
    {
      grade = 'B';
    }
    else if (marks >= 70)
    {
      grade = 'C';
    }
    else if (marks >= 60)
    {
      grade = 'D';
    }
    else if (marks >= 50)
    {
      grade = 'E';
    }
    else
    {
      grade = 'F';
    }
    System.out.println ("Grade is " +grade);
  }
}
```

output:- Javac IfElseDemo.Java
Java IfElseDemo
Grade is C.

→ Switch-case statement :-

The switch-case statement tests the value of given variable against a list of case values and when a match is found, a block of statements associated with that case is executed.

Syntax:-

```
switch (expression)
{
    case value1 : statements;
                  break;
    case value2 : statements;
                  break;
                  :
    default : statements;
}
```

Example:- Demonstrate switch-case statement.

```
public class SwitchDemo
{
 public static void main (String[] args)
 {
  int day = 2;
  switch (day)
  {
  case 1 : System.out.println ("sunday"); break;
  case 2 : System.out.println ("Monday"); break;
  case 3 : System.out.println ("Tuesday"); break;
  case 4 : System.out.println ("Wednesday"); break;
  case 5 : System.out.println ("Thursday"); break;
  case 6 : System.out.println ("Friday"); break;
  case 7 : System.out.println ("saturday"); break;
  default : System.out.println ("Invalid choice");
  }
}
}
```

o/p: Monday

* Loop statements :-

The Java supports following looping statements.

Those are
- while loop
- Do-while loop
- for loop

• While loop:

The Java while loop is used to iterate a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.

Syntax:-

```
while (condition)
 {
  // code to be executed
 }
```

• Do-while loop:

If the number of iteration is not fixed and you must have to execute the loop at least once, it is recommended to use do-while loop.

syntax:

```
do
 {
  // code to be executed
 } while (condition);
```

• for loop:

If the number of iteration is fixed, it is recommended to use for loop.

There are three types of for loop in Java.

- Simple for loop
- For-each loop
- Labeled for loop

- Simple For loop:

Syntax:

```
for(initialization; condition; incr/decr)
{
 //code to be executed
}
```

- Foreach loop:

Syntax:-

```
for(Type var: array)
{
 //code to be executed
}
```

- Labeled For loop:

Syntax:

```
Labelname:
for(initialization; condition; incr/decr)
{
//code to be executed
}
```

Examples:- While

```
public class WhileExample
{
public static void main(String[] args)
{
  int i=1;
  while(i<=10)
  {
    System.out.println(i);
    i++;
  }
}
}
```

o/p:- 1
2
3
4
.
.
.
10

Example: Do-While

```
public class DoWhileExample
{
public static void main (String[] args)
{
 int i=1;
 do{
    System.out.println(i);
    i++;
    }while(i<=10);
  }
 }
```

o/p: 1, 2, 3, ... 10

Example: For

```
public class ForExample
{
public static void main (String[] args)
{
 for(int i=1; i<=10; i++)
  {
   System.out.println(i);
  }
}
}
```

o/p: 1, 2, 3, ... 10

* Break and continue statements :-

→ The statements break and continue alter the normal control flow of compound statements.

→ The break statement immediately jumps to the end of the appropriate compound statement (loop).

→ The continue statements immediately jumps to the next iteration (if any) of the appropriate loop.

• break:-

When a break statement is encountered inside a loop, the loop is terminated and program control resumes at the next statement following the loop.

Syntax:- break;

Example:

```
public class BreakDemo
{
public static void main(String[] args)
{
 for(int i=1; i<=10; i++)
 {
   if (i==5)
   {
     break; // Terminate loop if i is 5
   }
  System.out.println(i);
 }
 System.out.println("Loop is over.");
}
}
```

output: C:\>Javac BreakDemo.Java

C:\>Java BreakDemo

1
2
3
4
Loop is over.

→ In simple words, the break keyword is used to breaks (stopping) a loop execution.

• Continue :-

When a continue statement is encountered inside the body of a loop, remaining statements are skipped and loop proceeds with the next iteration.

Syntax: continue;

Example:

```
public class ContinueDemo
{
  public static void main(String[] args)
  {
   for(int i=1; i<=10; i++)
    {
      if(i%2==0)
      {
        continue; // skip next statement if i is even
      }
     System.out.println(i);
    }
   }
}
```

output: C:\> Javac ContinueDemo.java

C:\> Java ContinueDemo

1
3
5
7
9

→ In simple words, The continue keyword is used to skip the particular recursion only in a loop execution.

* Simple Java Standalone programs :-

1. Fibonacci Series in Java

```
class Fibonacci
{
public static void main(String args[])
{
 int n1=0, n2=1, n3, i, count=10;
 System.out.print(n1+" "+n2); // printing 0 and 1
 for(i=2; i<count; i++)
 {
  n3=n1+n2;
  System.out.print(" "+n3);
  n1=n2;
  n2=n3;
 }
}
}
```

output: 0 1 1 2 3 5 8 13 21 34

2. prime Number

```
class primeNum {
    public static void main(String args[])
    {
      int num=17; //int num=Integer.parseInt(args[0]);
      int flag=0;
      for(int i=2; i<num; i++)
      {
        if(num%i==0)
        {
          System.out.println(num + " is not a prime");
          flag=1;
          break;
        }
      }
      if(flag==0)
        System.out.println(num + " is a prime number");
    }
}
```

output: 17 is a prime number

3. palindrome Number:

```
class palindrome
 {
 public static void main(String args[])
 {
  int r, sum=0, temp;
  int n=454; // n= Integer.parseInt(args[0]);
  temp=n;
  while(n>0)
  {
   r=n%10;
   sum=(sum*10)+r;
   n=n/10;
  }
  if(temp==sum)
  {
   System.out.println(" palindrome Number");
  }
  else
  {
   System.out.println(" Not palindrome");
  }
 }
}
```

output:-
palindrome Number

4. Factorial

```
class Factorial
 {public static void main(String args[])
 {
  int i,fact=1;
  int num=5; // n= Integer.parseInt(args[0]);
  for(i=1;i<=num;i++)
   {
    fact=fact*i;
   }
  System.out.println("Factorial of "+num+" is: "+fact);
 }
}
```

output:- Factorial of 5 is 120

5. Armstrong Number

```
class ArmstrongNum
{
public static void main (String args[])
{
 int sum=0, r, temp;
 int n=153; // n=Integer.parseInt(args[0]);
 temp = n;
 while (n>0)
 {
  r = n%10;
  sum = sum + (r*r*r);
   n=n/10;
 }
 if (temp == sum)
 {
   System.out.println ("Armstrong number");
 }
 else
  {
   System.out.println ("Not a Armstrong Number");
  }
}
}
```

output: Armstrong Number