JAVA CHAPTER 1:-DATA TYPE ,FLOW CONTROL AND ARRAY

```
//1.1= public static void main
public class point_1{
    public static void main(String[] args) {
     /*
        1|public:=It is an Access modifier, which specifies from where
                    and who can access the method.
        2|tatic:=The main() method is static so that JVM can invoke it
                    without instantiating the class
        3|void:=It is a keyword and is used to specify that a method
                doesn't return anything
        4|main:=It is the name of the Java main method. It is the
           identifier that the JVM looks for as the starting
      point of the java program. It's not a keyword.
        5|String[] args :=It stores Java command-line arguments and is
                    an array of type java.lang.String class. Here,
the name of the String array is args but it is not fixed and the user
can use name in place of it.
      */
    }
}
Point:-2
/*1.2= Data type
    |1.2.1-->primitive type and string
    |1.2.2-->literals & variable & Assignments
    |1.2.3-->blocks & variable scope
    |1.2.4-->java operator
 */
public class point_2{
```

```
    public static void main(String[] args) {
        /*
         primitive data type :=Java defines eight primitive types of
data: byte, short, int, long,
         char, float, double, and boolean. The primitive types are
also commonly referred to as simple
          types, and both terms will be used in this book. These can
be put in four groups:
            •    Integers This group includes byte, short, int, and
long, which are for whole-valued signed numbers.
            •    Floating-point numbers This group includes float and
double, which represent
            numbers with fractional precision.
            •    Characters This group includes char, which represents
symbols in a character set,
            like letters and numbers.
            •    Boolean This group includes boolean, which is a
special type for representing
            true/false values
        */


        /*
         Literals:=A literal is a fixed  value that we assign to a
variable in program
         int a=10;
         char b='A';
         in this 10 and A are literals
```

- thre are some type in literals they are following

1:integer literal =integer litrals are assigned to the variable of data type byte,shot,int,long

example:- byte a=23;

shot b=56;

int c=2673;

long d=366327647L;

2:float literals =used for data type float and double

example:- float f=34.6f;

double e=643.35;

3:char and string literals = used for char and string type

example:= char g='A';

String h="viraj";

4:boolean literals = used for boolean type like true and false

example:= boolean i=false;

bolllean j= true;

*/

/*

blocks & variable scope:=

variable

1|Local Variables:=

- A variable defined within a block or method or constructor is called a local variable.
- nitialization of the local variable is mandatory before using it in the defined scope.
- class main{

public static void main(String args[]){

int a=6; --> local variable

}

}

2| Instance Variables:=

- *Instance variables are non-static variables and are declared in a class outside of any method, constructor, or block.*
- *initialization of an instance variable is not mandatory. Its default value is 0.*
- *Instance variables can be accessed only by creating objects.*
- *class main{*

  public int a=6; --> instance variable

  public static void main(String args[]){

  }

3| Static Variables:=

- *Static variables are also known as class variables.*
- *Initialization of a static variable is not mandatory. Its default value is 0.*
- *static variables are declared using the static keyword within a class outside of any method, constructor or block.*
- *class main{*

  public static int a=6; --> static variable

  public static void main(String args[]){

  }

*/

/* Block:=

● Block refers to a set of statements inside 2 curly braces (one opening ‘{“ and one closing “}”).

Java supports 2 types of blocks. They are:

1|static block:=

● If the block of code is declared with the static keyword, it is called Static Block in Java.

● A static block only executes for a single time for the life cycle of the program.

● But the static block is always executed before the main method.

● public class main{

static{

//this is static block

}

public static void main(String args[]){

//program;

}

}

2|non- static block:=

● If you declare a block without any static keyword, then it is a Non-Static Block.

● A non-static may execute n number of times as it depends upon the user.

● class main{

{

```
                        //this is non-static block
                    }
                    public static void main(String args[]){
                        //program
                    }
                }


        */
       /*
         Operator:=
```

- Operator in Java is a symbol that is used to perform operations. For example: +, -, *, / etc.
- type of oprator :=

```
                    1|Assignment- = += -= *= /= %= &= ^= |= <<= >>= >>>=
                    2|Bitwise-  ^ | &
                    3|Logical- && || ?:
                    4|Arithmetic- * - + / % >> << >>>
        */


    }
}
Point :- 3
//this is not important
Point :- 4
/*1.4= Array
    |1.4.1-->Defineing and using Array
```

```
    |1.4.2-->Multydimentional Array
*/
class point_4{
    public static void main(String[] args) {
        //Defineing and using Array
          /*
           * Arrays are used to store multiple values in a single
variable, instead of declaring separate variables
           *  for each value.To declare an array, define the variable
type with square brackets
          */
          /*int a[]={1,3,4,5,7,8,};
          for(int c:a){
            System.out.println(c);
          }  */


          //Multydimentional Array
          /*
            1|A multidimensional array is an array of arrays.
            2|Multidimensional arrays are useful when you want to
store data as a tabular form, like a table
              with rows and columns.
            3|To create a two-dimensional array, add each array within
its own set of curly braces:
          */
          int [][] numbers={{5,6,7,8},{5,6,7,88,87,98}};
          for(int i=0;i<numbers.length;++i){
```

```
            for(int j=0;i<numbers[i].length;++j){
                System.out.println(numbers[i][j]);
            }
        }
    }
}
```