

JSpider

JAVA

JAVAC (JAVA COMPILER)

- It checks the given program to find syntax mistakes.
- If there is any syntax mistakes then java compiler throws an error.
- The errors which is throw by javac are called as compile time errors.
- The java compiler generates .class file if there are no syntax errors in the program.

JVM (JAVA VIRTUAL MACHINE)

- JVM is an interpreter which is going to
 - (i) Read one line of code.
 - (ii) Understand it.
 - (iii) Executes it.
- If JVM is not able to understand a line of code then JVM throws an error at runtime.
- The errors thrown by JVM are called as runtime errors or exceptions.

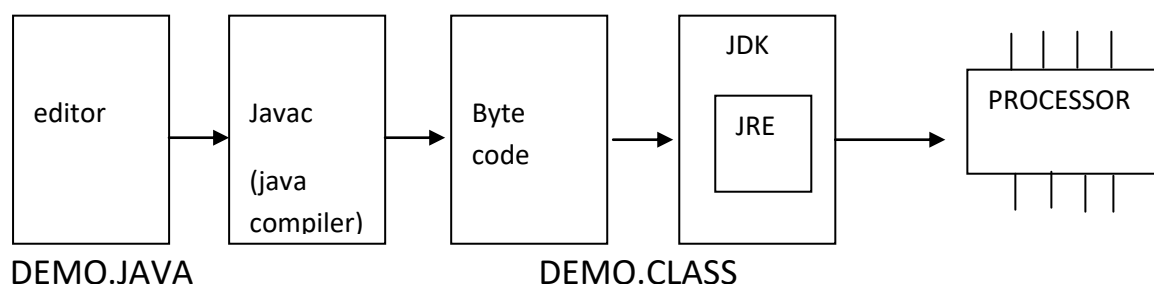
JRE (JAVA RUNTIME ENVIRONMENT)

- It is used to setup the environment in with the help of operating system to execute the program of JVM.

JDK (JAVA DEVELOPMENT KIT)

- It is a software package which contains java compiler, jvm and other necessary files which are required to compile and executes java programs.

JAVA WORK FLOW



CHAPTER 1 : KEYWORDS, IDENTIFIERS & VARIABLES

KEYWORDS

- They are reserved words which have a predefined meaning.
- It is not possible to change the meaning of keywords in any programming language.
- Keywords are used to define a class, declare a variables or define a method etc. in java.

Keywords in Java

Abstract	Continue	For	New	Switch
Assert ***	Default	Goto*	Package	Synchorinised
Boolean	Do	If	Private	This
Break	Double	Implements	Protected	Throws
Byte	Else	Imports	Public	Throws
Case	Enum****	Instanceof	Return	Transient
Catch	Extends	Int	Short	Try
Char	Final	Interface	Static	Void
Class	Finally	Long	Strictfp**	Volatile
Const*	Float	Native	Super	while

* not used

** added in 1.2

*** added in 1.4

**** added in 5.0

IDENTIFIERS :

- It is used to identify class or interface, methods, variables etc.

Rules of Identifiers :

- Identifiers can be alphanumeric.
- Identifiers should not start with numbers.
- \$ and _ (underscore) are the only two special character allow.
- An identifier can start with \$ and _.
- Keywords can not be used as identifiers.

VARIABLES :

- A variables is a named memory location which holds the value for the program.
- To use a variables we have to follow three steps :
 1. Declaration
 2. Initialization
 3. Utilization (Usage)

Declaration of a variables :

Syntax :

Datatype variablename;

Eg : int age;

- It is a statement which defines what type of data will be stored in the given variables.

Initialization of a variable :

Syntax :

Variable=value;

Eg: age=25;

- It is a statement which is written to store the value into the variable using assignment operator.

Utilization of a variables:

- Statements which are written to use the values of the variables are called as utilization.

Primitive Data types in java

				Default value
Integer	Byte	8 bits	1 byte	0
	Short	16 bits	2 byte	0
	Int	32 bits	4 bytes	0
	Long	64 bits	8 bytes	0l
Decimal	Float	32 bits	4 bytes	0.0f
	Double	64 bits	8 bytes	0.0

	Char	16 bits	2 bytes	Blank space
	Boolean	8 bits	1 bytes	false

CHAPTER 2 : OPERATORS

- It performs operations on the given operands and produce results.

Arithmetic Operator

- To perform any arithmetic operator
- $+$, $-$, $*$, $/$, $\%$

Note- Variables of the program should not be printed with “double quotes”.

Note – System.out.print – prints & keep cursor in same line.

System.out.println – Prints & makes the cursor to next line.

Formula to get how much a datatype maximum and min value range

$$-2^{n-1} \text{ to } 2^{n-1}-1$$

Eg : for bytes

$$-2^7 \text{ to } 2^7-1$$

$$-128 \text{ to } 127 \text{ (bytes)}$$

If we perform any arithmetic operation with the help of arithmetic operators with the help of arithmetic operators the result variables has to be decided by the following method.

Max(int, type of 1st operand, type of 2nd operand)

Program :

(a)

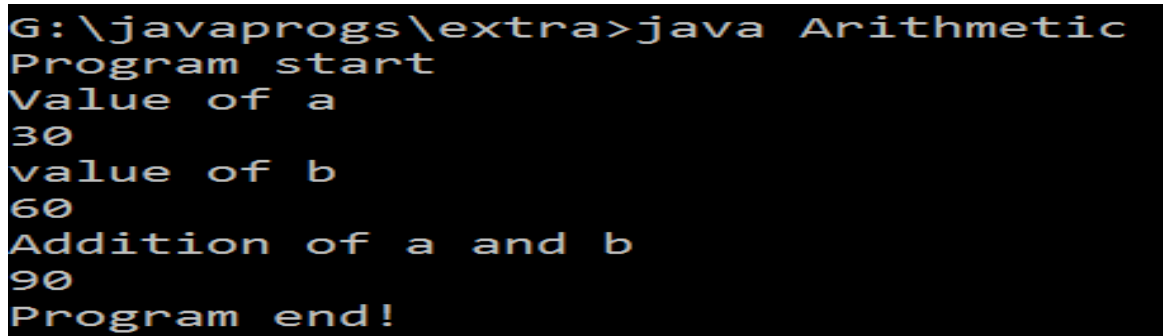
```
class Arithmetic
{
    public static void main(String[] args)
    {
        System.out.println("Program start");
        //declaration
        int a;
        int b;
```

```

int res;
//initialization
a=30;
b=60;
res=a+b;

        System.out.println("Value of a");
                System.out.println(a);
        System.out.println("value of b");
                System.out.println(b);
        System.out.println("Addition of a and b");
                System.out.println(res);
        System.out.println("Program end!");
    }
}

```



```

G:\javaprogs\extra>java Arithmetic
Program start
Value of a
30
value of b
60
Addition of a and b
90
Program end!

```

(b)

```

class Arithmetic2
{
    public static void main(String[] args)
    {

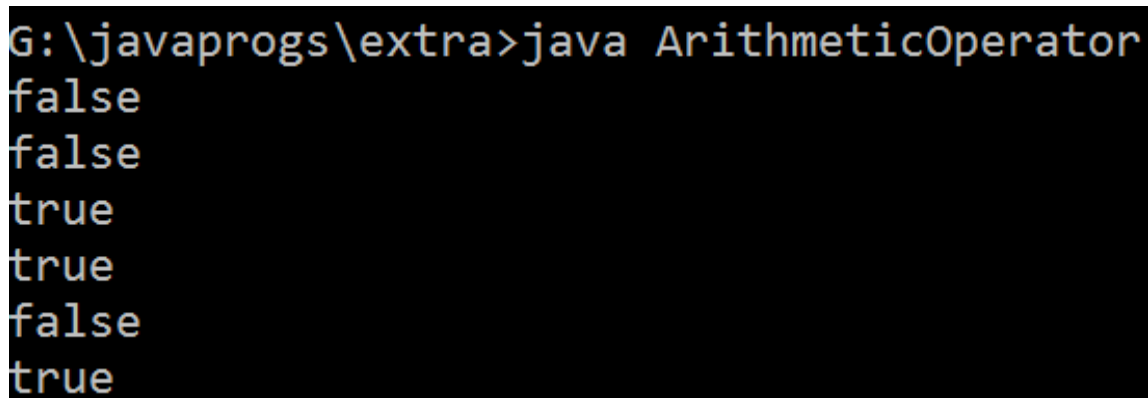
        int a=10; //declaration and initialization at same place
        int b=20;
        int c=a+b;

        System.out.println("value of a"+a);
        System.out.println("Value of b"+b);
        System.out.println("Addition of a and b"+c);
    }
}

```

(c)

```
class ArithmeticOperator
{
    public static void main(String[] args)
    {
        int a=5;
        int b=10;
        System.out.println(a>b);
        System.out.println(a>=b);
        System.out.println(a<b);
        System.out.println(a<=b);
        System.out.println(a==b);
        System.out.println(a!=b);
    }
}
```



```
G:\javaprogs\extra>java ArithmeticOperator
false
false
true
true
false
true
```

Concatination Operator

- It helps in concatenating (joining) a string value with any other value.

Combination of Concatination

- 2+ "hello" – 2hello
- "hello"+123 – hello123
- "hello" + "World" – helloworld
- 2+3+ "hello" – 5hello
- "hello"+2+3 – hello2+3 – hello23
- 2+ "hello"+3 – 2hello3

Increment & Decrement Operator :

- Those operators are used to increase or decrease the value of a variable by 1 units.

Increment Operator (++) :

- There are two types of increment operator.

1. Pre-increment.

2. Post-increment.

- If you write post or pre increment operators with a variables independently without any mathematical of expressions then, both operators will have same results.

Pre Increment	Post increment
First increment	Substitute
Substitute	Perform operation
Perform operation	Increment value

- Increment operator can not be used with Boolean datatypes.
- Any value can not be used directly with increment operators.
- Note :

```
Char a1= 'A';  
a1 = a1+1; //can not write like this  
a1=66;  
a1= 'B'
```

- **V.V.I notes**

Byte b1=10; B1++; S.o.p(b1);	Byte b1=10; B1=b1+1; s.o.p(b1);
Does not show error	Show error just because of max function

Program

(a)

```
class Increment  
{  
    public static void main(String[] args)  
    {  
        System.out.println("Program start");  
        int a1=10;  
        int b1=10;  
        int res;  
        System.out.println("a1 = "+a1);  
        System.out.println("b1 = "+b1);  
    }  
}
```

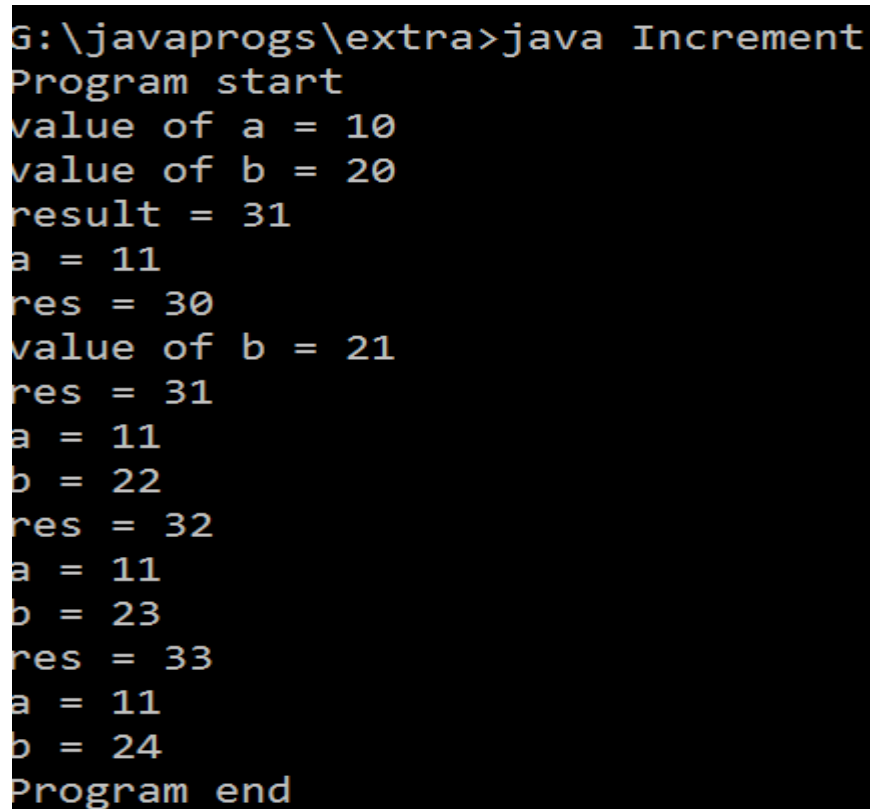
```

        res=++a1 + 10;
        System.out.println("res = "+res);

        res=b1++ + 10;
        System.out.println("res = "+res);
        System.out.println("a1 = "+a1);
        System.out.println("b1 = "+b1);

        res=b1++ + 10;
        System.out.println("res = "+res);
        System.out.println("a1 = "+a1);
        System.out.println("b1 = "+b1);
        res=b1++ + 10;
        System.out.println("res = "+res);
        System.out.println("a1 = "+a1);
        System.out.println("b1 = "+b1);
        System.out.println("Program end");
    }}

```



```

G:\javaprogs\extra>java Increment
Program start
value of a = 10
value of b = 20
result = 31
a = 11
res = 30
value of b = 21
res = 31
a = 11
b = 22
res = 32
a = 11
b = 23
res = 33
a = 11
b = 24
Program end

```


(b)

```
class Increment1
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
        int a1=10;
        System.out.println("a1 = "+a1);
        a1++;
        System.out.println( "a1 = "+a1);
    }
}
```

```
G:\javaprogs\extra>java Increment1
Hello World!
a1 = 10
a1 = 11
```

(C)

```
class Increment2
{
    public static void main(String[] args)
    {
        int a1=10;
        System.out.println(a1++);
        System.out.println(++a1);
    }
}
```

```
G:\javaprogs\extra>java Increment2
10
12
```

(d)

```
class Increment3
{
    public static void main(String[] args)
    {
        byte b1=10;
        b1++;
        System.out.println(b1);} }
```

```
G:\javaprogs\extra>java Increment3
11
```

Decrement Operator: (--)

- Those operators are used to decrease the value of a variable by 1 units.
- There are two types of decrement operator.
 3. Pre-decrement
 4. Post-decrement
- If you write post or pre decrement operators with a variables independently without any mathematical of expressions then, both operators will have same results.

Pre decrement	Post decrement
First increment	Substitute
Substitute	Perform operation
Perform operation	Increment value

- decrement operator can not be used with Boolean datatypes.
- Any value can not be used directly with decrement operators.
- Note :

```
Char a1= 'B';  
a1 = a1-1;  
a1=65;  
a1= 'A'
```

CHAPTER 3 : FLOW CONTROL STATEMENT

- It is a statement which is used to control the execution flow of a program.
- There are two types of flow control statements.
 1. Branching Statement (Decision making Statement)
 2. Looping Statement

Branching Statement (Decision Making)

- These statements are used to execute a group of statements based on a Boolean condition.
- The type of decision making are :-
 - (a) If statement
 - (b) If else statement
 - (c) If else if statement
 - (d) Switch case statement

IF STATEMENT

- If statement executes gives groups of statements within its body only if the Boolean condition result is true.
- Syntax :

```
If(condition) – true
{
    Statement ;
}
```

PROGRAM

```
class IfStatement
{
    public static void main(String[] args)
    {
        System.out.println("Program Start");
        int v1=50;
        if(v1>10)
        {
            System.out.println("v1 is greater than 10");
        }
        System.out.println("Program end");
    }
}
```

IF-ELSE STATEMENT

- Once the statements written in if block will be executed only if the Boolean condition is true and if Boolean is false then statements of else block will be executed.
- **Syntax:**

```
If(condition) true
{
    Statement;
}
Else
{
    Statement;
}
```

Program

```
class IfElseStatement
{
    public static void main(String... args)
    {
        int a=97;

        if (a>b)
        {
            System.out.println("print a");
        }
        else
        {
            System.out.println("print b");
        }
    }
}
```

IF-ELSE-IF STATEMENT

SYNTAX :

```
    If(Boolean condition 1)
    {
        Statement;
    }
    Else if(condition 1 )
    {
        Statement;
    }
    Else {}
```

Program

```
class Ifelseif
{
    public static void main(String[] args)
    {
        System.out.println("Program Start");
        int marks=25;
        if(marks>79 && marks<=100)
        {
            System.out.println("first class distinction");
        }
        else if (marks>=60 && marks<=79)
        {
            System.out.println("first class ");
        }
    }
}
```

```

    }
    else if (marks>=50 && marks<=59)
    {
        System.out.println("second class ");
    }
    else if (marks>=35 && marks <=49)
    {
        System.out.println("third class ");
    }
    else if (marks>100)
    {
        System.out.println("Invalid number");
    }
    else
    {
        System.out.println("You are fail");
    }
}
}

```

**** note :** logical operator are used to combine multiple conditions.

And operator (&&)				Or operator ()		
C1	C2	Result		C1	C2	Result
T	T	T		T	T	T
T	F	F		T	F	T
F	T	F		F	T	T
F	F	F		F	F	F

SWITCH CASE

- It is used whenever we have to compare the given value only equals (==) conditions.
- Switch case statements provides more readability for the program.
- It is not possible to compare the conditions other than equals conditions.
- Syntax :


```

Switch(choice)
{
    Case 1: Statement;
        Break;
    Case 2: Statement;
        Break;
    Default : Statement;
}

```
- Note : Break statement stop the execution of the given block at a given line of code. Writing break statement after default case is not mandatory in switch case statement.

Program

```
class Switch
{
    public static void main(String[] args)
    {
        System.out.println("Program Start");
        char a='C';
        switch(a)
        {
            case 'A': System.out.println("alphabet A ");
            break;
            case 'C' : System.out.println("aphabet C ");
            break;
            default : System.out.println("Invalid ");
        }
    }
}
```

LOOPING STATEMENTS

- They are used to perform repetitive task in the given program with lesser lines of code.
- Different types of loops are :-
 - (a) For loop
 - (b) While loop
 - (c) Do-while loop
 - (d) For each loop /advanced/enhanced loop

FOR LOOP (ITERATION) :

- One complete execution cycle of a loop is called as iteration.
- The number of iteration of a loop depends on the condition of the loop.
- Syntax :

```
For(initialization; condition; counter)
{
    Statement;
}
```
- For loop is a type of loop which is used whenever the logical start and logical end is well defined.

Basic Program :

```

Class Basic
{
    Public static void main (String ar[])
    {
        (1)      (2)      (4)
        For(int count=1; count<=3; count++)
        {
            (3)
            System.out.println("Hello World");
        }
    }
}

```

Op – Hello World
Hello World
Hello World

Iteration 1st – 1, 2, 3, 4

Iteration 2nd – 2,3,4,

Iteration 3rd – 2,3,4

Tracing :

Count=1	
Count<=3, 1<=3, true print hello world	
Count=2	
Count<=3, 2<=3, true print hello world	
Count=3	
Count<=3, 3<=3, true print hello world	
Count=4	
Count<=3, 4<=3, false terminate	

Prgm ; wap to print 1 to 5

```

Class Pgm2
{
    Public static void main (String ar[])
    {
        For(int i=1; i<=5;i++)
        {
            System.out.println(i);
        }
    }
}

```

Tracing

I=1
1<=5 true print i=1

I=2 2<=5 return true print i=2
I=3 3<=5 return true print i=3
I=4 4<=5 return true print i=4
I=5 5<=5 return true print i=5
I=6 6<=5 return false for loop terminate

Pgm : wap to print the numbers between 33 to 74 in descending order.

Class A

```
{
    Public static void main (String ar[])
    {
        For (int i=74; i>=33;i--)
        {
            System.out.println(i);
        }}}
```

Tracing

I=74 74>=33 true print i=74
I=73 73>=33 true print i=73
I=72 72>=33 return true print i=72
I=32 32>=33 return false and terminate

Prgm : wap to display only even no between 1 to 100?

```
For(int i=1; i<=10;i++)
{
    If(i%2==0)
    {
        S.O.P. (i);
    }
}
```

Tracing

I=1 I<=10, 1<=10 return true then if block execute and check the condition If(1%2==0) return false so it terminate and return back to for block
I=2

I<=10, 2<=10 return true then if block execute and check the condition If(2%2==0) return true and print 2 and condition goes to again for loop block
I=3 I<=10, 3<=10 return true then if block execute and check the condition If(3%2==0) return false so it terminate and return back to for block
I=4 I<=10, 4<=10 return true then if block execute and check the condition If(4%2==0) return true and print 2 and condition goes to again for loop block -----
I=11 I<=10, 11<=10 return false so the for block is terminate and program is end.

Pgm : wap to display number between 1 to 100 which are divisible by both 2 and 5.

```

For( int i=1; i<=100;i++)
{
    If(i%2==0 && i%5==0)
    {
        System.out.println(i);
    }
}

```

NESTED FOR LOOPS

- A for loop written within the body of another for loop is called as nested for loop.
- Syntax :

```

For(initialization; condition; counter)
{
    For(initialization; condition; counter)
    {
        Statement ;
    }
    Statement
}
}

```

Pgrm :

```

Class A
{
    Public static void main (String ar[])
    {
        For(int i=1;i<=3;i++)
        {
            System.out.println("outer loop");
            For(int j=1; j<=3;j++)
            {
                System.out.println("inner loop");} } }

```

Tracing

I=1
I<=3, 1<=3 return true print outer loop and goes to inner loop
J=1
J<=3, 1<=3 return true print inner loop
J=2
J<=3, 2<=3 return true print inner loop
J=3
J<=3, 3<=3 return true print inner loop
J=4
J<=3; 4<=3 return false inner loop terminate and outer loop execute again

I=2
I<=3, 2<=3 return true print outer loop and goes to inner loop
J=1
J<=3, 1<=3 return true print inner loop
J=2
J<=3, 2<=3 return true print inner loop
J=3
J<=3, 3<=3 return true print inner loop
J=4
J<=3; 4<=3 return false inner loop terminate and outer loop execute again

I=3
I<=3, 3<=3 return true print outer loop and goes to inner loop
J=1
J<=3, 1<=3 return true print inner loop
J=2
J<=3, 2<=3 return true print inner loop
J=3
J<=3, 3<=3 return true print inner loop
J=4
J<=3; 4<=3 return false inner loop terminate and outer loop execute again

I=4
I<=3, 4<=3 return false and terminate outer loop

Prgm : wap to print pattern


```
For(int i=1;i<=3;i++)  
{  
    For(int j=1;j<=3;j++)  
    {  
        System.out.print(" * ");  
    }  
    System.out.println(); // moves to next line  
}
```

Tracing

I=1
I<=3; 1<=3, return true , inner loop is executed
J=1
J<=3;1<=3 return true print *
J=2
J<=3;2<=3 return true print *
J=3
J<=3;3<=3 return true print *
J=4
J<=3;4<=3 return false and inner loop is terminated, condition again goes to outer for loop.

I=2
I<=3; 2<=3, return true , inner loop is executed
J=1
J<=3;1<=3 return true print *
J=2
J<=3;2<=3 return true print *
J=3
J<=3;3<=3 return true print *
J=4
J<=3;4<=3 return false and inner loop is terminated, condition again goes to outer for loop.

I=3
I<=3; 3<=3, return true , inner loop is executed
J=1
J<=3;1<=3 return true print *
J=2

J<=3;2<=3 return true print * J=3 J<=3;3<=3 return true print * J=4 J<=3;4<=3 return false and inner loop is terminated, condition again goes to outer for loop.
I=4 I<=3; 4<=3, return false, outer loop is terminated.

WHILE LOOP

Syntax :

```
While(Boolean_condition)
{
    Statement1;
}
```

- It is a type of looping statement. It is used whenever the logical start and logical end is not well defined.
- Pgm

```
Int count=1;
While(count<=5)
{
    System.out.println(count);
    Count;
}
```

DO-WHILE LOOP

Syntax :

```
Do
{
    Statement;
}
While(Boolean condition);
```

Prgm :

```
Int count=1;
Do
{
    System.out.println(count);
    Count++;
}
While(count<=5);
```