1.4.9. OPERATORS

Operator in java is a symbol that is used to perform operations. For example: +, -, *, / etc. There are many types of operators in java which are given below:

- Arithmetic Operator,
- Shift Operator,
- Relational Operator,
- Bitwise Operator,
- Logical Operator,
- Ternary Operator and
- Assignment Operator.
- **Java Operator Precedence**

Operator Type	Category	Precedence
Unary	postfix	*expr*++ *expr*--
	prefix	++*expr* --*expr* +*expr* -*expr* ~ !
Arithmetic	multiplicative	* / %
	additive	+ -
Shift	shift	<<>>>>
Relational	comparison	<><= >= instanceof
	equality	== !=
Bitwise	bitwise AND	&
	bitwise exclusive OR	^
	bitwise inclusive OR	|
Logical	logical AND	&&
	logical OR	||
Ternary	ternary	? :
Assignment	assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Java Unary Operator

The Java unary operators need only one operand. Unary operators are used to execute various operations i.e.:

- incrementing/decrementing a value by one
- negating an expression
- inverting the value of a Boolean

//Unary operator Example

```
class OperatorExample
{
```

```
public static void main(String args[])
{
int x=10;
System.out.println(x++); //10 (11)
System.out.println(++x); //12
System.out.println(x--); //12 (11)
System.out.println(--x); //10
}
}
```

Output:

10
12
12
10

Java Arithmetic Operators

Java arithmetic operators are used to perform various operations like addition, subtraction, multiplication, and division. They act as fundamental mathematical operations.

//Arithmetic operators

```
class OperatorExample
{
public static void main(String args[]){
System.out.println(10*10/5+3-1*4/2);
}
}
```

Output: 21

Java Assignment Operator:

Java assignment operator is one of the most general operators. It is used to allocate the value on its right to the operand on its left.

//Assignment operators

```
class OperatorExample
{
public static void main(String args[]){
int a=10;
int b=20;
a+=4;//a=a+4 (a=10+4)
b-=4;//b=b-4 (b=20-4)
System.out.println(a);
System.out.println(b);
}
}
```

Output:

14
16

Java Ternary Operator

Java Ternary operator is used as one liner replacement for if-then-else statement and used a lot in java programming. it is the only conditional operator which takes three operands.

// Ternary operator

```
class OperatorExample
{
public static void main(String args[]){
int a=2;
int b=5;
int min=(a<b)?a:b;
System.out.println(min);
}
}
```

Output:

2

1.4.10. .CONTROL FLOW

A control statements can be any one of given categories: selection, iteration, and jump.
Java's Selection Statements Java provides two selection statements: if and switch.
These statements permit you to control the flow of your code execution based upon situation known only during run time.

Selection:

If
If else
Ladder if
Nested if else
Switch statement

//if else example

```
public class IfElseIfExample {
public static void main(String[] args) {
  int marks=65;

  if(marks<50){
    System.out.println("fail");
  }
  else if(marks>=50 && marks<60){
    System.out.println("D grade");
  }
  else if(marks>=60 && marks<70){
    System.out.println("C grade");
  }
  else if(marks>=70 && marks<80){
    System.out.println("B grade");
  }
  else if(marks>=80 && marks<90){
    System.out.println("A grade");
```

```
  }else if(marks>=90 && marks<100){
     System.out.println("A+ grade");
  }else{
     System.out.println("Invalid!");
   }
}
}
```

Output:
C grade

//switch

```
public class SwitchExample2 {
public static void main(String[] args) {
   int number=40;
   switch(number){
   case 10: System.out.println("10"); break;
   case 20: System.out.println("20");break;
   case 30: System.out.println("30"); break;
   default:System.out.println("Not in 10, 20 or 30");
   }
}
}
```

Output:
"Not in 10, 20 or 30

Iteration:

- While
- do while
- for

//while

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

Output:1 2 3 4 5 6 7 8 9 10

//do-while

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

```
   int i=1;
   do{
     System.out.println(i);
   i++;

   }while(i<=10);
}
}
```

Output:1 2 3 4 5 6 7 8 9 10

//for

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

Output:1 2 3 4 5 6 7 8 9 10

jump:

Java provides three jump statement:

- **break**
- **continue**
- **return**

// break and continue

```
        publicclassBreakAndContinue{ publicstaticvoidmain(String args[]) { int[] numbers=
    newint[]{101,102,103,104,105,106,107,108,109,110};
                    intadd = 0;
        for(int i=0; i< numbers.length; i++){
                System.out.println("iteration: " + i);
        if(i == 5){
                    System.out.println("calling break statement");
                                        break;
                                }
        if(i%2 != 0){
        add = add + numbers[i];
                    System.out.println("calling continue statement");
                                        continue;
                }
                System.out.println("Last line of loop executed only for even number of iterations: "
        + numbers[i]);
             }
             System.out.println("This is outside the loop, sum: " + add);
          }
        }
```

Output:

Output is:
iteration: 0
Last line ofloop executed only for even number of iterations: 101
iteration: 1
calling continue statement
iteration: 2
Last line ofloop executed only for even number of iterations: 103
iteration: 3
calling continue statement
iteration: 4
Last line ofloop executed only for even number of iterations: 105
iteration: 5
calling break statement
This is outside the loop, sum: 206

1.4.11.ARRAYS

Array is a collection of contiguous or associated data items that share a general name.

Creating an Array

Arrays must be declared and created in the system memory before they are used

Creation of an array involves three steps

1. Declare the array
2. Create memory locations
3. Put values in to the memory locations.

Declaration of Arrays

Arrays in java may be declared in 2 forms

1. Type arrayname[];
2. Type[] arrayname;

Example

```
int number[];
int[] number;
```

Creation of Array

- After declaring an array we want to create memory.Java permits us to create arrays using new operator.

Syntax

arrayname=new type[size];

Ex:

number=new int[5];
average=new float[10];

Initialization of Arrays

- The initial step is to place values in to the array created. This process is called as initialization.

Syntax: arrayname[subscript]=value;

Ex: number[0]=10;
number[1]=20;

Syntax: type arrayname[]={list of values};
Ex: int number[]={10,20,30,40,50};

One Dimensional Array
A list of items can be known one variable name with only one subscript and a variable is known as a single subscripted variable.

Array Length

- All arrays store the allocated size in a variable named length
- Access the length of the array a using a.length

Ex:int asize=a.length;

Example Program: Sorting a list of numbers

```
class Sorting
{
public static void main(String args[])
{
int number[]={10,20,45,5,15};
int n=number.length;
System.out.print( "Given List”);
for(int i=0;i<n;i++)
{
System.out.print(" "+number[i]);
}
System.out.println("\n”);
for(int i=0;i<n;i++)
{
for(int j=i+1;j<n;j++)
{
if(number[i]<number[j])
{
int temp=number[i];
number[i]=number[j];
number[j]=temp;
}
}
}
System.out.println("Sorted List”);
for(int i=0;i<n;i++)
{
System.out.print(""”+number[i]);
}
System.out.println("");
}
```

```
}
```

Output:

Given List: 10 20 45 5 15
Sorted List:45 20 15 10 5

Two Dimensional Arrays

- A list of items can be known one variable name using 2 subscripts.
- That variable is called 2 dimensional array
  - First index=row
  - Second index=column

Ex: int myarray[][];
myarray=new int[3][4];

Example Program:

```
class Mutable {
final static int rows=20;
final static int columns=20;
public static void main(String args[])
{
int product[][]=new int[rows][columns];
int rows,columns;
System.outprintln("MUX table");
System.outprintln("");
int i,j;
for(int i=10;i<rows;i++)
{
for(j=10;j<columns;j++)
{
product[i][j]=i*j;
System.outprintln(""+product[i][j]);
}
System.outprintln("");
}
} }
```

Output:

100 110 120 130 190
110 121 132 143 209
190 209 228 247 361

Variable Size Arrays

- Java considers multidimensional arrays as arrays of arrays
- It is feasible to declare a 2 dimensional array as
- int x[][]=new int[3][];
- These create 2d array as having different lengths for each row X[0]=new int[2];

```
X[1]=new int[4];
X[2]=new int[3];
```