UNIT I – BASIC CONCEPTS **9**

Object Oriented Programming - Benefits of OOP – Applications of OOP. Java fundamentals: Features of java – Java development environment – Bytecode - Data types- Variables -Operators – Expressions – Functions – Static Members – Arrays – Strings – Classes and objects – Constructing objects using constructors.

JAVA TOKENS

Each individual smallest unit/element in a Java program is called a Java token. Tokens supported in Java include keywords, identifiers, variables, operators, constants, special characters, etc.

DATATYPES:

Java defines eight simple types of data: byte, short, int, long, char, float, double and Boolean. They can be put into 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 of representing true/false values.

These data types form the basis for all other types of data that you can create.

Integers:

Java defines four integer types- byte, short, int and long. All of these are signed, positive and negative values. Java doesn't support unsigned, positive-only integers. The width and ranges of these integer types vary widely, as shown in this table:

Name	Width	Range
long	54	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
int	32	-2,147,483,648 to 2,147,483,647
short	16	-32,768 to 32,767
byte	8	-128 to 127

long:

long is a signed 64 bit type and is useful for those occasions where an int type is not large enough to hold the desired value. The range is quite large. This makes it more useful when big, whole numbers are needed.

Eg: long a;

int:

It is a signed 32-bit type and it is the most commonly used data type. It is commonly used in control loops and to index arrays. The int type is the most versatile and efficient type.

Eg: int b;

short:

It is a 16-bit type. It is probably the least-used Java type, since it is defined as having its high byte first. This type is mostly applicable to 16-bit computers, which are becoming scarce.

Eg: short s;

byte:

The smallest integer type is byte. It is a signed 8-bit type. Variables of this type are useful when you're working with a stream of data from a network or file.

Eg: byte b,c;

Floating point types:

Floating point numbers are also known as real numbers; and are used when evaluating expressions that require fractional precision. There are two kinds of floating point types - float and double. float represents single precision numbers and double represents double precision numbers.

The width and ranges of floating point types are shown here:

Name	Width	Range
double	64	4.9e-324 to 1.8e+308
float	32	1.4e-045 to 3.4e+038

Float:

It specifies a single precision value that uses 32 bits of storage. Single precision is faster on some processors and takes half as much storage as double precision. The float type variables are useful when the values are either very large or very small.

Eg: float hightemp;

double:

Double precision is denoted by the double keyword It uses 64 bits to store a value. Double precision is faster on some processors than single precision.

All transcendental math functions such as sin(), cos(), sqrt() return double values.

char:

In java, the datatype used to store characters is char. Java uses Unicode to represent characters. Java char is a 16 bit type. The range of a char is 0 to 65,536. There are no negative chars. The standard set of characters known as ASCII ranges from 0 to 255.

boolean:

Java has a simple type, called Boolean for logical values. It can have only one of two possible values, true or false. This is the type returned by all relational operators, such as a<b This Boolean type is used by conditional expressions like if, for etc..,

---

VARIABLES:

Variables are the basic unit of storage in a java program. A variable is defined by the combination of an identifier, a type and an optional initializer. All variables have a scope, which defines their visibility and a lifetime.

Declaring a variable:

Every variable must be declared before they are used.

Syntax: type identifier [=value][,identifier[=value]..];

Type-name of class or interface

Identifier-name of the variable

Scope and lifetime of variables:

Java allows variables to be declared within any block. A block is begun with a curly brace and ended by closing curly brace. A block defines the scope of a variable. i.e., any variable declared in a block has its scope within the closing curly brace of that block.

Eg: a program to demonstrate block scope

```
class Scope
{
	public static void main(String args[])
	{
		int x; //known to all code within main
		x=10;
		if(x==10)
		{ //start new scope
			int y=20; //known only to this block x and y both are known here
			System.out.println("x and y:" +x + " " +y);
			x=y*2;
		}
		// y=100; // Error! Y is not known here
```

```
            // x is still known here
            System.out.println("x is" +x);
        }
}
```

OPERATORS

An operator is a symbol that takes one or more arguments and operates on them to produce a result. Operator tells the computer to perform certain mathematical or logical manipulations. Operators are used in programs to manipulate data and variables.

TYPES OF OPERATORS:

- Arithmetic operators
- Relational operators
- Logical operators
- Assignment operators
- Increment and decrement operators
- Bitwise operators
- Conditional operators
- Special operators

ARITHMETIC OPERATORS [+,-,*,/,%]

- Arithmetic operators are used to construct mathematical expressions.
- Java provides the basic arithmetic operators.
- These can operate on any built-in numeric data type of java

OPERATOR	MEANING
+	Addition (or) unary plus
-	Subtraction (Or) unary minus
*	multiplication
/	division
%	Modulo division(Remainder)

Arithmetic operators are used as shown below:

a-b a+b
a*b a/b
a%b -a*b

Here a and b may be variables or constant and are known as operands.

PROGRAM:

```
class Test
{
	public static void main(String args[])
	{
		int a=15,b=3;
		System.out.println("Add="+(a+b));
		System.out.println("Sub="+(a-b));
		System.out.println("Mul="+(a*b));
		System.out.println("Div="+(a/b));
		System.out.println("Modulo="+(a%b));
	}
}
```

OUTPUT:

Add=18
Sub=12
Mul=45
Div=5
Modulo=0

RELATIONAL OPERATORS:

- The comparisons can be done with the help of relational operators.
- The symbol '<' meaning 'less than' containing a relational operator is termed as relational expression.
- The value of relational expression is either true or false.

OPERATOR	MEANING
<	is less than
<=	is less than or equal to
>	is greater than
>=	is greater than or equal to
==	is equal to
!=	is not equal to

PROGRAM:

```
class Test
```

```
{
	public static void main(String args[])
	{
		int a=5,b=3;
		if(a>b)
		{
			System.out.println("A is greater");
		}
		else
		{
			System.out.println("B is greater");
		}
	}
}
```

OUTPUT:

A is greater.

LOGICAL OPERATORS:

- The logical operators && and || are used to form compound conditions by combining two or more relations.
- An expression combining two or more relational expressions is termed as logical expression.
- It also yields a value of true or false, according to the truth table.

Op-1	Op-2	Op-1&&Op-2	Op-1||Op-2
True	True	True	True
True	False	False	True

PROGRAM:

```
class Test
{
	public static void main(String args[])
	{
		int a=4,b=2;
		System.out.println("res="+(a&&b);
	}
}
```

OUTPUT:res= 0

ASSIGNMENT OPERATOR:

Assignment operators are used to assign the value of an expression to a variable. The symbol is " = ".

EXPRESSION	MEANING
X +=y	X=x+y
X - =y	X=x-y
X *=y	X=x*y
X /=y	X=x/y
X=y	X=y

PROGRAM:

```
class Assign
{
	public static void main( String args[])
	{
		int a=1,b=2,c=3;
		a+=5;
		b*=4;
		c+=a*b;
		c%=6;
		System.out.println("a="+a);
		System.out.println("b="+b);
		System.out.println("c="+c);
	}
}
```

OUTPUT:

a=6
b=8
c=3

INCREMENT AND DECREMENT OPERATOR:

- The increment(++) and decrement (--) operators can be prefixed or postfixed
- The ++ or – can appear before or after the value it increments or decrements

PROGRAM:1[increment]

```
class IncrementOperator
{
	public static void main(String args[])
	{
		int m=10,n=20;
		System.out.println(“m=”+m);
		System.out.println(“n=”+n);
		System.out.println(“++m=”+ ++m);
		System.out.println(“n++=”+ n++);
		System.out.println(“m=”+m);
		System.out.println(“n=”+n);
	}
}
```

OUTPUT:

```
m=10
n=20
++m=11
n++=20
m=11
n=21
```

PROGRAM:2[decrement]

```
class DecrementOperator
{
	public static void main(String args[])
	{
		int m=10,n=20;
		System.out.println(“m=” +m);
		System.out.println(“n=” +n);
		System.out.println(“--m=” + --m);
		System.out.println(“n--=” + n--);
		System.out.println(“m=” +m);
		System.out.println(“n=” +n);
	}
}
```

OUTPUT:

m=10
n=20
--m=9
n--=20
m=9
n=19

CONDITIONAL OPERATOR:

- The character pair ?: is a ternary operator available in java.
- The operator is used to construct conditional expressions of the form
- If(exp1) ? exp2 : exp3;
- This can be achieved using if ....else statement

PROGRAM:

```
class Test
{
        public static void main (String args[])
        {
                int a=4,b=3;
                if(a>b) ? System.out.println("A=" +a); : System.out.println("B="+b);
        }
}
```

OUTPUT:

A=4

BITWISE OPERATOR:

- Java has a distinction of supporting special operators known as bitwise operators for manipulation of data at bit level.
- These operators are used for testing the bits, or shifting them to the right or left.
- These operators may not be applied to float or double.

OPERATOR	MEANING
&	Bitwise AND
!	Bitwise OR
^	Bitwise exclusive OR
~	One's complement

<<	Shift left
>>	Shift right
>>>	Shift right with zero fill

PROGRAM:

```
class Test
{
	public static void main(String args[])
	{
		int a=4;
		System.out.println("a="+<<a);
		System.out.println("a="+<<<a);
	}
}
```

OUTPUT:

a=001
a=000

SPECIAL OPERATOR:

Java supports some special operators of interest such as instanceof operator and member selection operator(.)

1. Instanceof operator:

The instanceof is an object reference operator and returns true if the object on the left-hand side is an instance of the class given on the right-hand side. This operator allow us to determine whether the object belongs to a particular or not.

EXAMPLE:

Person instanceof student → is true if the object person belongs to the class student; otherwise it is false.

2. Dot operator

The dot operator (.) is used to access the instance variables and methods of class objects. It is also used to access classes and sub-packages from a package.

PROGRAM:

```
class Test2
{
	public static void main(String args[])
	{
		Test1 t1=new Test1();
```

```
                t1.add();
        }
}
class Test1
{
        int a=10,b=20,c;
        add()
        {
                System.out.println("c="+(a+b));
        }
}
```

OUTPUT:

C= 30
