Chapter 5: Variables & Expressions

The topics

1. What is a Variable?

1.1 Variable

A variable is a **memory location** that holds a particular value within the program and can change it during the execution of the program. The value stored in a variable is known as constant or literal. First, we need to declare any variable before using it. Declaration of a variable requires the data type of the variable and a name of the variable.

1.2 Rules to be followed while naming a variable in Java (Note : this naming is to be followed for all types of identifiers – variable, class, object)

i. Variable names can contain alphabets, digits, dollar and underscore.
ii. Variable names should start with any alphabet or underscore or dollar.
iii. Variable names cannot start with digits.
iv. Variable name cannot have any keywords
v. Variable name cannot contain any special characters or symbols.
vi. Variable name cannot have any space in between i.e., it should be one word.
vii. In Java, variable names are case-sensitive.

1.3 Initialization of variables

Variable need to be declared and initialized within the program.
Variables are declared in the following manner –

<Data type> variable_name;

i. declare an integer variable named marks
int marks;
ii. declare a String variable named name
String name;
iii. declare a character variable named grade
char grade;

Variables are initialized in the following manner –

variable_name = value;

i. store 68 in marks
marks = 68;
ii. assign “Rahul” to name
name = “Rahul”;
iii. assign ‘A’ to grade
grade = ‘A’;

1.4 Scope of a variable

It generally refers to the program region within which a variable is accessible. The scope of a variable can be of two types, as local variable and global variable.

2. What is a Constant?

A constant is a value given to the program which is fixed. We can make a variable as constant using the keyword **final**. By this, the value of the variable will not be changed during the program execution.
Eg. final double pi=3.142;   here adding the final keyword in front of the variable declaration makes the value as constant.

Advantages of Constants:

1. They make the program easier to read and check for correctness.
2. If a constant needs to be changed, all it required to do is the change in the declaration.

3. What is an Expression?

A valid combination of operator and operand that computes and gives a result is known as an expression. In Java, an expression consists of variables and constants with operators that operate on them.
e.g.   a = b + c;
Area = l * b (area of rectangle);
s = u * t + 0.5 * t * t ;   (distance, $s = u \times t + \frac{1}{2}at^2$)

3.1 Types of expressions

- **Arithmetic**
- **Relational (Conditional)**
- **String**
- **Single expression**
- **Compound expressions**.

3.1.1 Arithmetic Expression

An Arithmetic Expression is an expression that contains arithmetic operators (+, –,*, / , %) in it. Arithmetic expressions can be either **Pure expressions** or **Mixed (impure) expressions**. In a pure expression, all the operands are of the same data type, whereas in a mixed expression, the operands are of mixed or different data types. Also, it can be an **Integer expression** or a **Real expression.** Integer expressions are formed by connecting integer constants and integer variables with arithmetic operators, whereas real expressions use real constant and/or variables.

3.1.2 Relational Expression

A Relational Expression is an expression that contains relational operators (<, >, <=, >=, ==, !=) apart from arithmetic operators in it. It is a conditional statement that results true or false.

Example:
A+B ***Arithmetic expression***
A>B ***Relational expression***
(A+B) / (A-B) Arithmetic expression (i.e. +, - and / all are arithmetic operators)
(A+B) > (A-B) Arithmetic and relational expression (+ and – are arithmetic and > is relational operator)

3.1.3 Compound Expression

If two or more expressions are combined within a pair of braces { }, then they become compound expressions.

3.2 Type Conversion

The process of converting one pre-defined type into another is called type conversion.
Two forms of type conversion are –

i. Implicit type conversion, where data types are promoted to higher data type present in the expression. Implicit conversion is done automatically by the compiler to prevent loss of data in an expression. It is also called COERCION.
   1. byte will be converted to char/int
   2. char will be converted to int
   3. int will be converted to long/float
   4. float will be converted to double
   5. double will be converted to long double

ii. Explicit type conversion, here an operand of a specific type is forcefully converted to another type. Explicit conversion is a forceful conversion done by the programmer in an expression. It can be done to promote a data type to a higher type (to prevent loss of data) or to convert to some lower type (adjust a higher value into a lower type variable). It is also called as TYPE CASTING.
   Syntax – (type) expression

```
e.g.     int a=5, b=6, c=8;
    float avg= (float) (a+b+c)/3;        // type promotion
    int n=25;
    double sum = sum + (double) n/2; //type promotion (conversion to higher type)
    int a = (int) Math.sqrt(4);          //type demotion (conversion to lower type)

  class Student
  {       public static void main()
          {       String name="Rahul Singh";
                  int roll_no=45;
                  int cls=8;
                  char sec='D';
                  double mrks=98.5;
```

```
                    System.out.println("Name of the student:"+name);
                    System.out.println("Roll number:"+roll_no);
                    System.out.println("Class:"+cls+" Section:"+sec);
                    System.out.println("Marks scored:"+mrks);
          }
}
```

Identify the tokens in the above program code:

Keywords – class, public, static, void, String, int, char, double, System

Identifiers – variables: name, roll_no, cls, sec, mrks
Class: Student
Methods: main( ), println( ) [both of them are pre-defined method]

Literals – "Rahul Singh", 45, 8, 'D', 98.5, "Name of the student:", "Roll number:",
"Class:", "Section:", "Marks scored:"

Punctuators – { }, ( ), ;

Operators – = (Assignment), + (Concatenation), .(dot)

4. *Operator & Operands*

The operations (specific tasks) are represented by **operators** and the objects of the operations are referred to as **operands**. Operators are the symbols that trigger a specific action upon the operands and the operands are the variables or constants upon which the operator operates.

4.1 Types of operator

1. **Arithmetic operator :**
These are the operators used for arithmetic, calculations like addition, subtraction, multiplication and division, e.g. (+, –,*, /, %)
% (MODULUS OPERATOR) – It returns the remainder part of any division.
/ (DIVISON OPERATOR) – It returns the quotient part of any division.

Integer Division	Fractional Division
5/3 = 1 (Q)	5.0/3.0 = 1.66666667 (Q)
3/5 = 0	3.0/5.0 = 0.6
5%3 = 2 (R)	5.0%3.0 = 2.0 (R)

10%2 = 0 (R) (it implies that 10 is completely divisible by 2 and 2 is a factor of 10)
10%3 = 1 (R) (it implies that 10 is not completely divisible by 3 and 3 is not a factor of 10)

2. **Arithmetic assignment operator :**
These operators are formed by combining one arithmetic operator with an assignment operator.
e.g. +=, – = , *= , /= , %=
Let us take an example, int a=5, b=7;
a += 2 is equivalent to a = a+2 (i.e. a=5+2 = 7)

a * = 3 is equivalent to a = a*3 (i.e. a=7*3 = 21)
b -= a is equivalent to b = b – a (i.e. b=7-21 = -14)
b /=5 is equivalent to b = b/5 (i.e. b=-14/5 = -2)

3. **Relational operators :**
In the terms relational operator, relational refers to the relationship's values can have with one another. These operators are used to create or build relational (conditional) statements. There are 6 relational operators for comparing numbers & characters. They are –
< (less than),
<= (less than or equal to),
> (greater than)
>= (greater than or equal to),
= = (equal to)
!= (not equal to)
Relational operator returns Boolean values (true/false) as output of any relational statement. E.g.
7>6 (true)
6<3 (false)
5+3==9-1 (true)
5>=3 (true)
5>=5 (true)

What is the difference between == and =?
== is a relational operator – it compares two operands for equality and returns true or false
= is an assignment operator - it assigns the value from R.H.S. operand to L.H.S. operand

4. **Logical operator :**
These operators are used to construct more complex decision-making expressions with relational operators. Logical operators are used to join more than one condition. Three logical operators in Java are **&& (AND), || (OR), ! (NOT).** Logical operators combine two or more relational statements together to form a single and compound relational expression.

The logical **OR** operator combines 2 or more expressions which evaluates to true if either of its operands evaluates true. E.g.
(A==B || B==C || C==A) it implies whether from A, B, C any two values are equal or not.

The logical **AND** operator combines 2 or more expressions into one. The resulting expression has the value true only if both of the expressions are true. E.g.
(A==B && B==C) it implies that whether A, B and C all are equal or not.

The logical **NOT** operator, a unary operator, reverse the truth table value of the expression.
!(A==B) it implies, if A is equal to B then the NOT operator will change the answer to false.

5. **Increment/decrement operator (++ /– –) :**
Increment operator (++) adds 1 to its operand and decrement operator (- -) subtracts 1 from its operand. Both increment and decrement operators come in two forms –
   i. **Prefix notation (++a / – –a)**
   ii. **Postfix notation (a++ / a– –)**

These operators are unary operators, because they act upon single operand.
Both the two forms have same effect upon the operand, but they differ when they take place is an expression. Note: a++ or ++a means a=a+1 (i.e. the value of the variable a will be increased by 1)

Prefix operator	***Postfix operator***
1. The prefix version comes before the operand i.e. ++ a or - - a	1. The postfix version comes before the operand i.e. a++ or a - -
2. In the prefix form, the increment/decrement operation is performed prior to assignment within the expression.	2. In the postfix form, the increment/decrement operation is performed after the assignment within the expression.
3. The pre-increment operation, such as ++a, increases the value of a by 1 before it is used for computation.	3. The post-increment operation, such as a++, uses the current value of a for computation first then increases it.

```
int a=5, b=6,c;
(i) c=++a * ++b;
Step1. ++a => a=6           Pre-increment
Step2. ++b => b=7           Pre-increment
Step3. c=a*b =6*7 = 42      Computation

(ii) c=a++*b++;
Step1. c=a*b = 5*6 = 30     Computation
Step2. a++ => a=6           Post-increment
Step3. b++ => b=7           Post-increment

(iii) c=a++*++b;
Step1. ++b => b=7    Pre-increment
Step2. c=a*b = 35    Computation
Step3. a++ => a=6    Post-increment

(iv) c=++a*b++;
Step1. ++a => a=6    Pre-increment
Step2. c=a*b = 36    Computation
Step3. b++ => b=7    Post-increment
```

```
Method of solving:
Step1. All the pre-operation (++/--)
Step2. Computation
Step3. All the post-operation (++/--)
```

```
Practise:
int a=5,b=6,c=7,d=8,e;
(i) e=a++ + --b - ++c + d--;
(ii) e=--a * b-- * ++c * d++;
(iii) e= a-- * ++b / c++ * --d;
```

6. **Shift operator :**

These operators perform a bit manipulator on data by shifting the bits of its right operator right or left.
>> Right shift
<< Left shift
E.g. 13 >> 1, the binary representation of 13 is 1101 is 4 bit the result of the shift operation is 0110, i.e. 6.
10<< 1, the binary representation of 10 is 1010 is 4 bit the result of the shift operation is 0100, i.e. 4

7. **Bitwise operator :**
The bitwise operators in Java are: **& (bitwise AND), | (bitwise OR), ^ (bitwise XOR), ~ (bitwise complement)**. The bitwise operators work with integral types i.e. byte, short, int and long types. The bitwise operations calculate each bit of their results by comparing the corresponding bits of the 2 operands on the basis of these 4 rules –
   i. For AND operations,1 AND 1 produces 1, any other combination produces 0.
   ii. For OR operations, 0 OR 0 produces 0, any other combination produces 1.
   iii. For XOR operations, 1 XOR 0 and 0 XOR 1 produces 1, any other combination produces 0.
   iv. The complement operation inverts the value of each bit of the operand.

8. **Conditional operator (? :) :**
This is a ternary operator & requires 3 operands.
The general form of the conditional operator –
    Expression1? Expression2: Expression3
If expression 1 evaluates to true, then the value of the whole expression is the value of expression 2, otherwise the value of the whole expression is the value of expression 3.

9. **Assignment operator (=) :**
This operator assigns the value of one operand to another operand or a constant to a variable. E.g. int a = 5; (it means 5 is being assigned to the variable a)
int b = a; (it means the variable b is assigned with the value that is stored in a)
int c = a+b; (First values of a and b are added and then it is finally stored in c)

10. **Special operators :**
   i. **new**, it is used to create a new object of a new array.
   ii. **instanceOf**, is used to determine whether its first operand is an instance of its second operand or not.
   iii. (**type**), it casts (converts) a value of one type to another type.
   iv. (**params**), it is used to delimit a comma-separated list of parameters.
   v. **[ ]**, it is used to declare arrays, create arrays and access array elements.
   vi. **.** (Dot) it is used to form qualified names.

5. Operator precedence & Operator associativity

5.1 Operator precedence

It determines the order in which expressions are evaluated. This in some cases, can, determine the overall value of the expression. In general, increment & decrement are evaluated before arithmetic operation, arithmetic expression is evaluated before comparisons, & comparisons are evaluated before logical expression. Assignment expression expressions are evaluated last.

Operators according to precedence

1. [ ], ( ), { }	parenthesis, braces
2. ++, --	increment / decrement operator
3. * , / , %	Arithmetic operator
4. + . –	Arithmetic operator
5. << , >>	Shift operator
6. < , > ,>= , <=	Relational operator
7. = = , !=	Relational operator
8. && , || , !	Logical operator
9. ?:	conditional operator
10. += ,–= ,*= , /=	Arithmetic assignment operator
11. =	Assignment operator

5.2 Operator Associativity

Associativity rules determine the grouping of operands & operate in an expression with more than one operator of the same precedence. When the operations in expression all have the same precedence rating, the associativity rule determines the order of the operations. For most operators, the evaluations are done left to right.

Example: x = a + b – c; here addition & subtraction have the same precedence rating and so a & b are added first, then c will be subtracted from the sum. Parenthesis can be used to overrule the default associativity. E.g. x = a + (b – c), in this case, b-c will be done first and then the summation.

X = A*B/C*D, in this case, A*B will execute, then (A*B)/C will execute, then (A*B/C)*D and finally assignment operation will execute.

X = A*(B/C)*D, in this case at first B/C will execute, then A*(B/C) and then (A*(B/C))*D will execute

Multiple question answer:

1. **Java is a .... language.**

A. weakly typed
B. strongly typed
C. moderate typed
D. None of these Answer: **Option B**

2. **How many primitive data types are there in Java?**

A. 6
B. 7
C. 8
D. 9 Answer: **Option C**

3. **In Java byte, short, int and long all of these are**

A. signed
B. unsigned
C. Both of the above
D. None of these Answer: **Option A**

4. **Size of int in Java is**

A. 16 bit
B. 32 bit
C. 64 bit
D. Depends on the execution environment Answer: **Option B**

5. **The smallest integer type is .... and its size is .... bits.**

A. short, 8
B. byte, 8
C. short, 16
D. short, 16 Answer: **Option B**

6. **Size of float and double in Java is**

A. 32 and 64
B. 64 and 64
C. 32 and 32
D. 64 and 32 Answer: **Option A**

7. **Determine output:**

```
class A{
        public static void main(String args[]){
                int x;
                x = 10;
                if(x == 10){
```

```
                int y = 20;
                System.out.print("x and y: "+ x + " " + y);
                y = x*2;
            }
            y = 100;
            System.out.print("x and y: " + x + " " + y);
        }
}
```

A. 10 20 10 100
B. 10 20 10 20
C. 10 20 10 10
D. Error

Answer: **Option D**

8. Automatic type conversion in Java takes place when

A. Two types are compatible and size of destination type is shorter than source type.
B. Two types are compatible and size of destination type is equal of source type.
C. Two types are compatible and size of destination type is larger than source type.
D. All of the above

Answer: **Option C**

9. Which of the following automatic type conversion will be possible?

A. short to int
B. byte to int
C. int to long
D. long to int

Answer: **Option C**

10. What is the output of the following program?

```
class A{
        public static void main(String args[]){
            byte b;
            int i = 258;
            double d = 325.59;
            b = (byte) i;
            System.out.print(b);
            i = (int) d;
            System.out.print(i);
            b = (byte) d;
            System.out.print(b);
        }
}
```

A. 258 325 325

B. 258 326 326
C. 2 325 69
D. Error

Answer: **Option C**

Solved questions

1. Write Java Statements for the following :
    a. Declare a character variable and assign 12 to it.
    b. Declare a double variable and assign the result of 1200 *8/100.
    c. Declare a variable to assign false to it.
    d. Declare a variable to hold the product of two variable x and y.

Ans:- a. int a =12; b. double dbl = 96; c. boolean b = false; d. int z = x * y;

2. Consider the following statements :
    int x =55, y= 10, z = x %y;
    What is the value of z after the execution?

Ans : **5**

3. Consider the flowing statements
    int a = 15, b=10,
    b = a++,
    what is the value stored in b after program executed?

Ans:- **16**

4. Define Class variable and instance variable.

Ans:- A variable declared within a class is an **instance variable**. The compiler creates a copy of the each of the instance variable of the class at runtime.

Example :
```
class sample
{
        int marks;
}
```

Class variables are declared using the modifier static. Memory is allocated once when the variable is encountered first within the class. The same copy of the class variable is shared with all instances.

Example:
```
class Sample
{
        static int x;
        public int x ();
        {       return x;
        }
        public void serX (int new X)
        {       x = new X;
        }
```

}

5. Classify the following literals into their appropriate data types

a.	987	b.	2569817	c.	25
d.	10	e.	true	f.	\n
g.	"Amar"	h.	1-800-999-888	i.	253.2

Ans: a. short b. long c. float d. byte e. boolean
f. char g. String h. string i. double

6. Difference between rules and conventions.

Ans :

Rules are defined by the programming language. Any violation leads to error messages. Unless the errors are corrected, the program cannot be executed.
Conversions are the general format followed by all programmers. A uniformity is maintained by whoever writes the program. It is an unwritten rule. Though the conventions are violated. The programs do not throw error messages. The programs can be executed.

7. Assuming centimeter is declared as int cm, write a java statement to convert centimeter to inch. One inch = 2.54 cm.

Ans:- double inch = cm/2.54;

8. Define post-decrement and pre-decrement operators.

Ans:- **Post Increment operators** – is given after the operand. Example : x --. It performs the decrements after the associated operations and then decrements the values of the operand by 1.
Pre-decrements operator – is given before the operand. Example –x. in this case the value of the operand is decremented before it performs any operations associated with the operand.

9. Name the different types of bitwise logical operators.

Ans:- & (bitwise AND), | (logical OR), ^ (bitwise exclusive OR), ~ (bitwise complement)

10. Name the operators

Ans:- & (bitwise AND), | (logical OR), ^ (bitwise exclusive OR)

Short answer questions:

1. Write down the rules for naming identifiers.
2. Write down the naming conventions of identifiers.
3. What are literals? Name and explain all the types of literals present in Java language.
4. Explain with examples the following:
    a. Global and local variables
    b. Static and non-static variables
    c. Dynamic initialization of a variable
    d. Use of **‘new’** keyword
5. Explain the working of bitwise operators
6. Explain the working of shift operators
7. What are pure and impure expressions?
8. Explain implicit and explicit conversion in one example only.
9. What is the significance of main( ) method in a Java program?
10. What is the significance of **‘String args[]’** in main method?
