Data types and Variables

* Data types -

Data type is used to specify the type of data that a variable can hold.

→ Variables are ment to store data.

→ Data is stored only after the variable is declared.

→ Data can be both sign and unsigned.

→ Any information is called as data.
Eg:- name, age, marks etc.

* There are two types of data type in java -

1) Primitive data types (Basic data types)
These data types are categories into four parts -

a) Integral - Have any numerical value without decimal point.

→ Integral data type is divided depending on the size :

Name	Size	Range	Default
- Byte	1	-128 to 127	0
- short	2	- 32768 to 32767	0
- int	4	- 2147483648 to 2147483647	0
- long	8	—	0

b) Floating-Point - Have numerical value with decimal point.

Type	Size	Range	Default
Float (6 to 7 digits supports after decimal)	4	$\pm 1.4E^{-45}$ to $\pm 3.4E+38$	0.0f
double (14-15)	8	$\pm 439E-324$ to $\pm 1.7E+308$	0.0d

c) char -

Size - 2 bytes

Range - 0 to 65535

default - \u0000

- To support unicode, java takes 2 bytes of size of a char.

d) boolean -

Size - undefined

default - false.

- True is represented using 'true'.
- False is represented using 'false'.

* For writting other languages (Hindi, Urdu), we use unicode.
* For English, ASCII is used, which is also a subset of unicode.

2) Reference data types -

a) array : a collection of data elements of the same type.

b) classes : a user-defined data type that defines set of properties and method.

c) interfaces : a user-defined data type that defines a set of methods without implementation.

* Variables -

A variable is a named container that stores a value. It is used to hold data that can be manipulated and used in a program.

Eg:- int a = 5;

(Here 'a' is a variable name)

* Types of variables -

i) Instance variable -

Instance variable are defined within a class, but outside of any method. They are associated with an instance of a class and can be accessed using the object of the class.

ii) Local variables -

These variables are defined within a method or block. They are temporary variables and their scope is limited to the method or block in which they are defined.

iii) Static variables - (Class variables)

Static variables are declared with the static keyword in a class but outside a method, constructor or a block.

* Rules to naming variables -

- Variable names differ in upper and lower cases.
- Variables names can contain alphabets, numbers and _, $.
- It can't start with a digits.
- It may start with an alphabet or, _ , $.
- It should not contains built-in class name as well as built in words like int, char.
- Name can be as long as possible.
- Follow camel cases (Mixed type)

  Eg:- rollNumber , avgMarks

* Java Literals -

A literal is a constant values (number) that is stored into a variable directly in the program.

→ Literals are written based upon the data types.

Eg:- 1) Z = 5 * x + 7 + y

Here 5 and 7 are literals (integer)

2) int value = 25 ;

Here 25 is the literal (integer)

3) long x = 125

(Here 125 is a integer literal and we are storing it in a 'x' variable which is long type.)

4) long x = 99999999999999999 ;

Here it is integer literal but it is too large and beyond of the integer size. So it shows error.

long x = 99999999999999999L ;

(Now it is right)

Types of Literals -

1) int literal -

Representing integer type. A number without decimal point is called int literal.

Eg:- int x = 10;
short y = 1255;

Here 10, 1255 are int literal.

2) float and double literals -

A number with decimal point.

- By default in java, any decimal value is treated as double literals.

Eg:- float f = 12.56f ; (If we don't write 'f' then it will considered as double by default)

- So always use 'f' in float type value.

float f = 12.56;
↙
Invalid

(It will show error, that lossy conversion from double to float.)

double d = 12.56; // Valid

double d = 12.56d; // valid

3) Char literals -

Representing character in single quotes.

Eg:- char ch = 'a';

4) String literals -

Representing character in double quotes.

Eg:- String str = "Hello";

5) Boolean literals -

It represents only two values true and false.

Eg:- boolean a = true;

* Note :

'int literals' can be represented in different number system.

Decimal - 10

Binary - 0b1010

Octal - 012

Hexadecimal - 0XA

Operator and Expressions

* **Operator** -

An operator is a symbol or keyword that performs an operation or a set of operations on one or more operands.

* Types of operators -

1) Arithmetic Operators -

These are used for performing mathematical operation.

Operator	meaning	example
+	Addition	A + B
-	Subtraction	A - B
*	Multiplication	A * B
/	Division	A / B
^	Power	A^3
%.	Reminder	A %. B

2) Assignment Operators -

An assignment operators is used for assign a value to a variable. The most common assignment operator is '='.

Eg:- assign value 5 to the variable x :

```
x = 5
```

* The statement " C = A + B " means that add the value stored in variable A and B then assign/store the value in variable C.

* +=, -=, /=, *=, %= these operators are known as compound operator and it is also called shorthand notation.

Eg:- x = x + 10; can be replaced by

→ x += 10

3) Relational Operators -

These operators which are used to compare or check the relation b/w two or more quantities.

Operator	Meaning	Example
<	less than	A < B
<=	less than or equal to	A <= B
==	Equal to	A == B
!=	Not equal to	A != B
>	greater than	A > B
>=	greater than or equal to	A >= B

4) Logical Operators -

These are used to combine multiple conditions.

→ AND (&&) -

Eg:- A < B && B < C

(Result is true if both A<B and B<C are true else Result will become false.)

→ OR (||) -

Eg:- A < B || B < C

(Result is true if either A<B or B<C are true else false.)

→ NOT (!) -

Eg:- ! (A > B)

(Result is true if A > B is false else false.)

5) Increment and Decrement operator - (Unary)

- Increment operators are the unary operators used to increment or add 1 to the operand value.

* The increment operator is denoted by the "++"

- Decrement operator is the unary operator, which is used to decrease the original value of the operand by 1.

* It is represented as the "--".

* ++, -- can be used as -

  → ++pre : Pre increment

  → post++ : Post-increment

  → --pre : Pre-decrement

  → post-- : post decrement

* In pre-increment/decrement first the value is incremented/decremented and then utilized.

* In post-increment/decrement first the value is utilized and then incremented/decremented.

Eg:- 1) int i, j = 2;

```
i = ++j;   // i = 3 , j = 3
```

2) int i, j = 2;

```
i = j++;   // i = 2, j = 3
```

6) Bitwise Operators -

These operators works on bits of data only and it ~~works~~ use only integer type of data.

- → & (AND)
- → | (OR)
- → ~ (NOT)
- → ^ (XOR)
- → << (Left shift)
- → >> (Right shift)
- → >>> (Unsigned right shift)

*

bit1	bit2	bit1 & bit2
0	0	0
0	1	0
1	0	0
1	1	1

Bitwise AND(&)

*

bit1	bit2	bit1 | bit2
0	0	0
0	1	1
1	0	1
1	1	1

Bitwise OR(|)

*

bit1	bit2	bit1 ^ bit2
0	0	0
0	1	1
1	0	1
1	1	0

Bitwise XOR(^)

*

bit	~bit
0	1
1	0

Bitwise NOT (~)

* AND(&) -

Eg:- int x = 10, y = 6, z;

x → 0 0 0 0 1 0 1 0

y → 0 0 0 0 0 1 1 0

---

z = x & y → 0 0 0 0 0 0 1 0 → 2 Ans

* OR(|) -

x → 0 0 0 0 1 0 1 0

y → 0 0 0 0 0 1 1 0

---

z = x | y → 0 0 0 0 1 1 1 0 → 14 Ans

* NOT(~) -

int x = 5;

x → 0 0 0 0 0 1 0 1

∴ ~x → 1 1 1 1 1 0 1 0

This is a negative value, to know its value, we have to convert it into 2's complement.

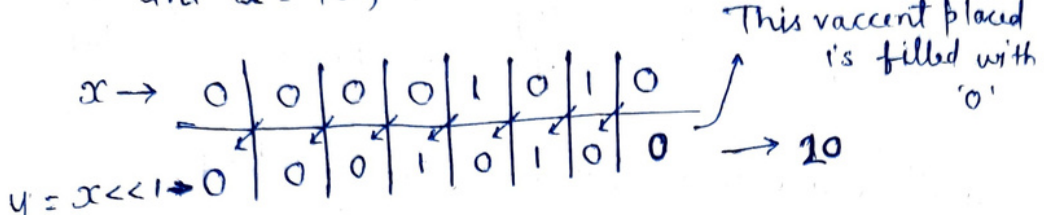
1's → 0 0 0 0 0 1 0 1

+ 1

2's → 0 0 0 0 0 1 1 0 → -6 Ans

* Left shift (<<) -

int x = 10 → n;



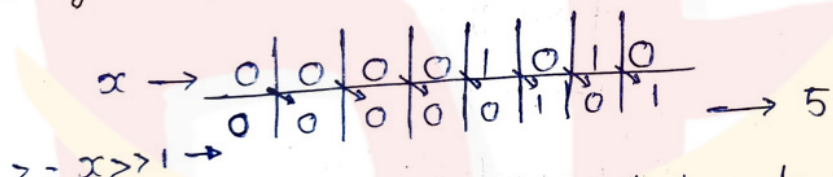
(Bits are shift by 1 space to left)

∴ y = x << 1 (n → x, K → 1)

∴ y = 20 $(n * 2^K)$ → formula

$= 10 \times 2^1 = 20$

* Right shift (>>)



(Bits are shifted by 1 space to right)

∴ z = x >> 1

z = 5 $\left(\frac{n}{2^k} = \frac{10}{2^1} = 5 \right)$

* Type Casting in java :

Type casting is the process of converting a value of one data type to another data type.

→ There are two types of type casting -

1. Implicit type casting (Widening or Automatic type casting)

This occurs when the conversion takes place automatically by the compiler.

Eg:- When an ~~it~~ int is assigned to a long, the conversion is done automatically.

Example-

```
int x = 125;
float y;
y = x;
System.out.println("Value of y" + y);
```

output - Value of y: 125.0

* In this casting, smaller size data type is always stored into larger size data type.

* It is also known as upcasting.

2. Explicit type casting (Narrowing or Manual type casting)

This occure when the conversion is done manually by the programmer. For example, when a double is assigned to an ~~it~~ int, the programmer needs to explicitly cast it as int.

Example-

```
double d = 10.5;
int i = (int) d;
System.out.println("Value of d:" + d);
System.out.println("Value of i:" + i);
```

Output- Value of d = 10.5
Value of i = 10

* In this casting, larger size data type is stored into smaller size data type.
* Due to difference in size it may lead to loss of data.

* Concatenation in java -

Concatenation is the process of joining two or more things to create a new string. This can be done using the "+" operator, which combines the string on both sides.

Eg:- `int x = 10, y = 20;`

→ `System.out.println(x + y);`

o/p = 30

→ `System.out.println("Number is" + y);`

o/p = Number is 20

→ `System.out.println(x+y + "Sum");`

o/p = 30 Sum

→ `System.out.println("Sum" + x + y);`

o/p = Sum 1020 (Here concatenation occures)

→ `System.out.println("Sum" + (x+y));`

o/p = Sum 30

→ `System.out.println("Sum of" + x + "and" + y + "is" + (x+y));`

o/p = Sum of 10 and 20 is 30

* How to read data from keyboard -

Java provides a class named Scanner which is used to take input from the user during runtime through the keyboard.

→ To use the 'Scanner class' in java, we first need to create an object of the Scanner class and pass the input source to it.

Eg:- `Scanner sc = new Scanner(System.in);`

'sc' is the reference of Scanner class

This is associated with keyboard.

* Scanner is a class which present in the 'java.util' package.

* There are various methods of the Scanner class to read data from the input source.

- nextInt() : For taking integer value as input.
- nextFloat() : For taking float value as input.
- nextDouble() : For double type value
- nextByte() : Byte type values
- nextShort(); Short type values
- nextLong() : Long type values
- nextBoolean() : Boolean values (true, false)
- next() : String value without space (single word)
- nextLine() : String value with space (Multiple words)

Example :

```
import java.lang.*;
import java.util.*;
Class kybRead
{
    Public static void main(String arg[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter your name");
            String name = sc.nextLine();

        System.out.println("Enter your roll");
            int roll = sc.nextInt();

        System.out.println("Enter your percentage");
            float Pc = sc.nextFloat();

        System.out.println("Enter your Contact");
            long cn = sc.nextLong();

        System.out.println("Enter your married
                                    status");

            boolean st = sc.nextBoolean();

        System.out.println("Enter your gender");
            String g = sc.next();
```

```
System.out.println("Details entered by you are:");
System.out.println("Name: " + name);
System.out.println("Roll :" + roll);
System.out.println("Percentage: " + Pc);
System.out.println("Contact: " + cn);
System.out.println("Status :" + st);
System.out.println("Gender: " + g);
}
}
```

* If we did not give the value properly while asking during runtime, we will get "InputMismatch-Exception".

* We don't have any method called as nextChar() to take character as input rather we have a method like:

  next().charAt(Index) → to take character input.

Eg:-
```
char gender;
S.o.p("Enter your gender");
gender = sc.next().charAt(0);
S.o.p(gender);
```

O/P - Enter your gender - Male

Here charAt(0) = M, so

It will print only 'M'.