DATA TYPE & OPERATORS

14/10/2022

Incrementaion, Decrementation operator

PART-2

⇒ Real Number:-

⇒ By defaut decimal value treated as Double. If you have to specify to the compiler to treat it as float. we need to suffix it with 'F' or 'f'.

```
float a=10.5; // CE: Possibly loss of precession
float a=10.5f; or float a=10.5F; // (valid)
```

NOTE:- Datatypes are actually represented to the compiler and jvm using reserved words.
Reserve words are normally in "Lowercase".

→ To map primitive data as object in java from JDK1.5 concept of "Wrapper class" was introduced.

e.g. → byte ----> Byte (class)
(Premitive)
Short ----> Short (c)
int -----> Integer (c) .....

⇒ Character: (char)

Size:- 16 bits (2 byte), minvalue:- , maxvalue = ?

⇒ to store any type of character char datatype is used; eg:- A, B, @, #, a, b

→ memory only understand 0, 1, By this type means binary form

1 bit

A → 0
B → 1

(2^1) → 1 bit
char

1 bit 1 bit

A 0 0
B 0 1
C 1 0
D 1 1

(2^2) → 2 bit (required)
char

1 bit 1 bit 1 bit

A → 0 0 0
B → 0 0 1
C → 0 1 0
D → 0 1 1
E → 1 0 0
F → 1 0 1
G → 1 1 0
H → 1 1 1

(2^3) → 3 bit (Required)
char

So, how many character increase the number of bits will increase.

⇒ ASCII (American standard code for information Interchange)

→ The American discovered total 128 character present in English language world of computer

128 ⇒ 2^7 ⇒ 7 bits + 1 bits ⇒ total 8 or 8 bits ⇒ 1 Byte
char char

So, in C we use only bound of 128 char: char → 1 Byte (in C)

→ But, if we see entire world and analyse all the language we found total char as - 65536 (char/symbol) like - English, Hindi, Bengali, marathi, ... (all)

⇒ Unicode (UTF) Transformation Format: (UTF-16)

IEEE → found 65536 char/symbols → 2^16 ⇒ 16 bits ⇒ 2 Bytes

⊛ So, Java follows unicode (UTF) format. This why char in java is 2 Byte ⇒ 2 (8) bits = 2 × 8 bits = 16

⇒ In ASCII or UTF :— 128 letters are same (English)

English letter (U)		Decimal value		English letter (L)
A	→	65	97	← a
⋮		⋮	⋮	⋮
Z	→	90	122	← z

⇒ Syntax of char :— use '' for write char.

```
char a = 'A'; //valid.
char a = "A"; // invalid.
```

⊛ ⇒ Truncation / Rounting to zero ? ⇒

```
int a = 25;
int b = 2;
int c = a/b; // c = 12
S.o.p = (c);
```

> float c = a/b; // c = 12.0 → why ? But ans should be 25/2 = (12.5) ?

Where → any expression does result → (int)
Real → any exp. ⇒ that type

int * int = int
int / int = int
float / float = float
double / double = double

{ 0 * 0 = NaN (output)
0 / 0 = NaN (Not a Number)

new ⇓
{byte * byte = int
int * int = int
long * long = int} > float
{float * float = float
double * double = double}

⇒ Whenever you are doing a expression in between any some (where) data type result will store only in Integer data type irrespective of where you are storing in float or etc.
e.g: int * int = int , int / int = integer (not give float)

⇒ You can store full by type casting. or store it in integer

⊛

(follows Base 2 format)				(IEEE) → single and double precision	
1 Byte	2 Byte	4 Byte	8 Byte	4 Byte	8 Byte
byte	short	int	long	float	double

2 Byte char } UTF (unicode format)

(IEEE) = International Electrical & Electronics Engg.

format ⇓

2 | 45 — 1
2 | 22 — 0
2 | 11 — 1
2 | 5 — 1
2 | 2 — 0
(101101)

⊛ ⇒ Type casting :— / Numeric Promotion :— / Implicit conversion :—

→ This happens when the two data types are compatible and also when we assign the value of a smaller data type to a larger datatype.

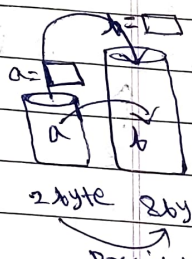
→ Type casting take place during the program design by programmer (explicit) or compiler itself (implicit).

⇒ There have two type of "Type casting" :—

(1) Implicite Type castion :— / Type conversion (in C) :— This conversion is done by the compiler itself.

When more than one data type of variables are used in an expression, the compiler converts data types to avoid loss of data.

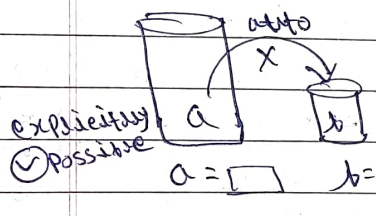
ex:- int n = 23;
char p = 'c';
float q = 2.88f;
n = n + p; // implicit type casting happen (char to int)
S.O.P(n); // 90
q = q + n; // 92.88 (I.T.C happen int to float)
S.O.P(q);



(1) This is also called as type conversion in C Langu

(2) Explicit type casting:-
This conversion is done by user. This is also known as typecasting. Data type is converted into another data type forcefully by the user.
Syntax:- (type) expression;
ex:- float a = 5.55;
int S = (int)a + 1; // Explicitly float to int type casting Done

⇒ Reverse type casting not possible automatically.



(i) Two type should be compatible
(ii) Source destination should be greater than current destination
int ⇒ float ✓ float ⇒ int ✗
(iii) Have to do explicitly casting.
(iv) By explicitly loss of precession might happen.

(*) ⇒ operators:-

int a = 5 (+) 10 ⇒ 15 (operator)

For performming any expression we use some symbols to do some certain operation. that is called operators. like-> +, -, /, * ...

⇒ Incrementation ? || Decrementation ?

```
int a=5;
a=a+1;   } incrementing the existing value by 1.
5+1
(6)
```

or `a++;//(a=a+1)`

```
int a=5;
a=a-1;
5-1
(6)
```

or `a--;//(a=a-1)`

⇒ (pre ⇒ ++a , Post ⇒ a++)

⇒ So, a++ (first use it's value or give it/ print it then increament it (Post))

++a (first increase the value then use it. (Pre))

a-- (first use it's value then decreament it (Post))

--a (1st decreament the value then use it (Pre))

ex:-
```
int a=5;
S.o.P(a); //a=5
a++;
S.o.P(a); // a=6
S.o.P(a++); // Print a=6
```
But actual will a=7

ex:-
```
int b=6;
S.o.P(b); //b=6
++a;
S.o.P(b); //b=6
S.o.P(++b); // Print b=7
```
actual value=7

ex:-
```
int a=5;
int b=a++;   // a=5 given to b, a=6
S.o.P(a); // a=6
S.o.P(b); // b=5
```

ex:-
```
int a=5
int b=++a;   // a=5
S.o.P(a); // a=6
S.o.P(b); // a=6
```

ex:-
```
int a=5; int b;
b = a++ + a++ + ++a;   //a=8
    5   + 6   + 8 ⇒
S.o.P(a); ?
S.o.P(b); ?
```

ex:-
```
int a=5; int b;
b = ++a + a++ + ++a + a--;
    (6  + 6   + 8   + 8 = 28)
S.o.P(a); // a=7
S.o.P(b); // 28
```

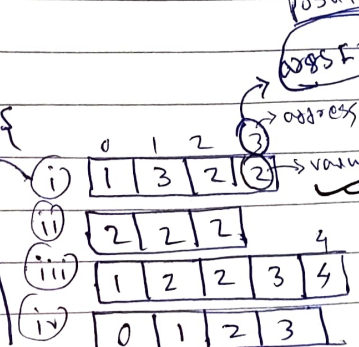
⇒ Try code snippets (Date - 14/10/2022)

```
Public class myFive{
    Public S V main(String[] args){
        String arg1 = args[1]; //3
        String arg2 = args[2]; //2
        String arg3 = args[3]; //2
        S.o.P("Arg is" + arg3);
```

Ans match

⇒ Which command line arguments should you pass to get Arg is 2?

Option :- (1) java myFive 1 3 2 2
(2) java myFive 2 2 2
(3) java myFive 1 2 2 3 4
(4) java myFive 0 1 2 3



(*) NOTE:- increment & decrement op. is applicable only for integral, floating type and character type not for boolean type. and also for variable not in boolean.