Operators, Datatypes and Variables

Operators

+ , - , * , / , % , ++ , --

Add op

subtract op

Multiply op

Divide operator

Modulo/Modulus op

Increment op

Decrement op

2+2 = 4

Post Increment

Pre Increment

Post Dec

Pre Dec

System.out.println(2+2) ⇒ 4

SOUT (3-2) ⇒ 1

SOUT (3*2) ⇒ 6

SOUT (3/2) ⇒ 1 ⇒ Integer/Integer

3/2 = 1.5 ⇒ floor Value : upper

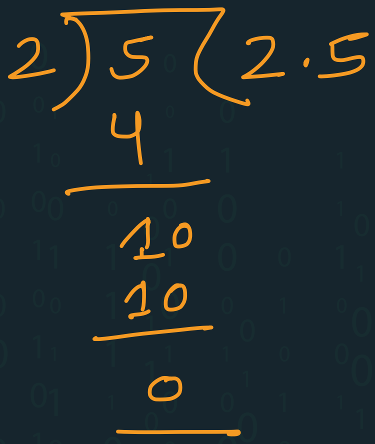
ceil value : lower Int

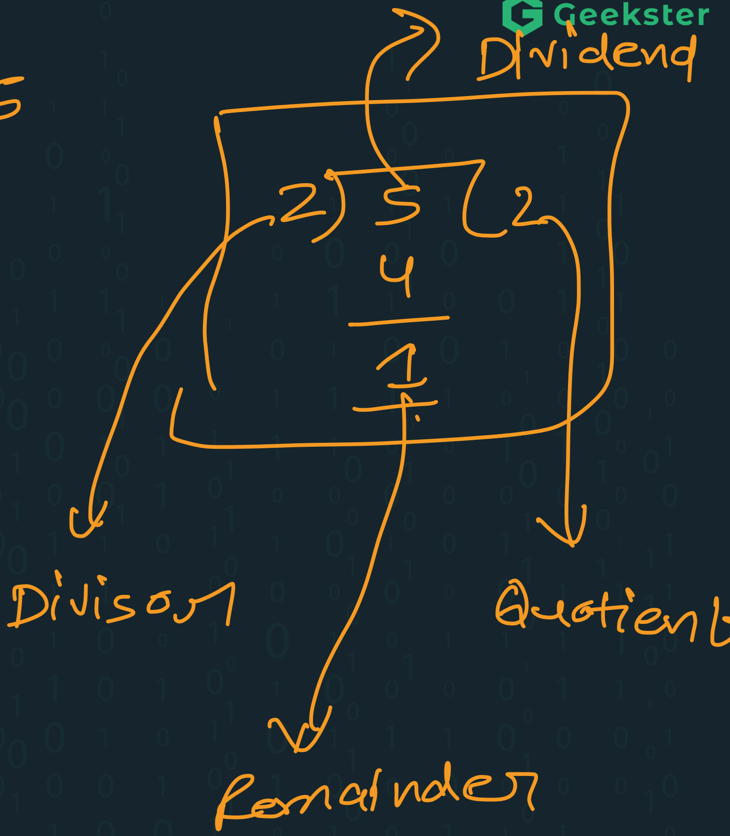
SOUT (5/2) ⇒ 2

SOUT (10/5) ⇒ 2

% op gives the remainder.

5/2 ⇒ 2 (Remainder = 1)





5 % 2 = 1

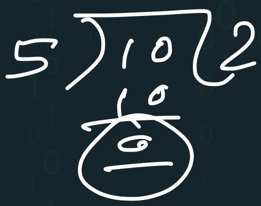
Dividend Divisor Remainder

5 / 2 = 2

Dividend Divisor Quotient

SOUT (3%2) ⇒ 1 → Rem

SOUT (10%5) ⇒ 0 → Rem.



floor value ⇒ Lower Integer value

ceil value ⇒ upper Integer value

3/2 ⇒ 1.5

2)3(1.5
2
10
10
0

floor(1.5) ⇒ 1
ceil(1.5) ⇒ 2

System.out.println(3/2) ⇒ 1

System.out.pln(3.0/2) ⇒ 1.5

System.out.pln(3/2.0) ⇒ 1.5

syst-- pln(3.0/2.0) ⇒ 1.5

If any of the value between dividend & divisor is in decimal form, then you will get decimal op

++ : Increment operator

2++ ⇒ 3

3++ ⇒ 4

System.out.println($(\overset{\nearrow}{2}++)$); ⇒ Error

Machine doesn't understand the $\boxed{2}$ here

bcoz it expects a variable at this place

but we are providing value.

-- : Decrement operator

2-- ⇒ 1

1-- ⇒ 0

SOUT (2--) ⇒

Variable Datatypes

Variable is a placeholder
↓
Box

Integer

→ SOUT(3)

→ SOUT(3*2)

→ SOUT(3x ——-)

format:-

int Dabba/Box Name = 3;

int → Type of placeholder/ Box/dabba

Dabba/Box Name → Name of PH

3 → value we are assigning in PH

int myVariable = 5;

we have defined a variable

Declaration	Initialization
`int myVar;`	`myVar = 3;`

# for proper code execution:

* first do the declaration and then do the initializ^{n}.

Declaration + Initializ^{n}

→ Definition

```
int myVar = 2;
```

Defin^{n}

```
myVar = 3;
int myVar;
```

✗ Incorrect

```
int myVar;
myVar =3;
```

```
int myVar;
System.out.println(myVar);
```

# DataTypes

Datatypes

Primitive DT

1.) byte
2.) short
✓ 3.) int
4.) long
✓ 5.) boolean
✓ 6.) float
✓ 7.) double
✓ 8.) char

Takes Integer Values

User Defined DT

1.) class
2.) structure
⋮ ⋮

Derived DT

1.) String
2.) Array

int ⇒ Represents Integer value

Boolean ⇒ True Or false
$\overset{1}{\text{True}}$ Or $\overset{0}{\text{false}}$
Yes or No
Haan or Na

float ⇒ Represents decimal value

double ⇒ " " "

char ⇒ " single alphabet. or special character like /, +, -, *, %, ?, !,

```
                // *** Playaround with code ***


// Players


// https://github.com/geeksterin/FS-15


// "static void main" must be defined in a public class.
public class Main {
    public static void main(String[] args) {
        char myChar = 'd';
        System.out.println(myChar);
        int myInt = 4;
        System.out.println(myInt);
        boolean myBool = false;
        System.out.println(myBool);
        // float myfloat = 1.2f;
        double myDouble = 23.87;
        System.out.println(myDouble);


        // int myVar; // Declaration
        // myVar=4;  // Initialization
        // int myVar = 4; // Definition = Declaration + Inititilazation

        //Post Inc and Pre Inc
        // int abc = 4;
        // System.out.println(abc++); // Post Increment => This operator is written after the variable name and the variable value will be used first and then it will get incremented.

        // System.out.println(myVar); // Pre Increment => This operator is written before the variable name and the variable value will be increment first and then it will get used.


//           System.out.println(++abc);

//           System.out.println(abc++);

//            System.out.println(abc);

        //           System.out.println(abc++);


//           System.out.println(abc++);

//           System.out.println(abc+10);
    }
}
```

// L-14 : 4
// L-15 : 5
// L-18 : 6

//   after L-18, the value of abc has become 7,

// L-20 : 7+10 = 17

// Variable Declaration : [DataType of Variable] [Name of Variable]; => int abc;
// ham machine ko batate hai ke is naam ka or is type ka variable exist krta hai, to ise samjho
// We tell the machine that this type of variable and this is the name of variable that exists, and with declaration, you can understand what is this variable

// Variable Inititialization : [Variable Name] = [Value]; => abc = 4;

// Thumb Rule =>
//   * Neither we can redeclare a variable (we cant use the same name of variable in the code again), nor we can redefine it. We can only reinitialize the variable. And the final value of the variable will be the one which is assigned at last.

// Floor Value is the nearest small integer
// Ceil Value is the nearest greater integer

// 2.3 => Floor value (2.3) => 2
//   Ceil value (2.3) => 3

// ; => Semicolon
// : => Colon

// HW:

// 1. Use pre and post increment operator on a variable at the same time.
// 2. Try to play around with Pre and post decrement operator
// 3. Find sum and difference of x and y. Take any value in x and y.
// 4. Find the area (pi*r*r) and perimeter (2*pi*r) of a circle. Take any values for pi and r.