Conditionals and loops + Calculator Program :

Conditions : any statement we check if it is true/ false and the condition have to be either true or false.

if statement :

syntax —

```
if (condn) {
    // body
}
```

// here condn will be either true or false,
   if true, then body will get implement
   if false, " " " not " "

if-else statement :

syntax ———

```
if (condn1) {
    // body1
}
else {
    //body2
}
```

// here if condn1 is true , then body1 will implement
   " " " false, " body2 " "

→ eg of if-else statement:

```
int salary = 25400;
if (salary >10000) {
        salary = salary + 2000;
} else {

        salary = salary + 1000;

}

System.out.println(salary);
```

```
27400
```

Multiple if-else statement:

Here more than one else condition will be there (condition other than if will be more than one).

→ eg of multiple if-else statement:

```
if (salary <10000){
        salary = salary + 1000; }
else if (salary = = 10000){
        salary = salary + 2000;
}
else {
        salary = salary + 3000;
}
```

if does not fall

then will fall here

// any case other than ~~first two~~ two condition will fall in the third condition (else part)

Loops : something which goes on doing same thing one after another.

suppose we need to print "Hello" 20 times then instead of writing sout 20 times, we can use loops which will iterate 20 times (or run 20 times)

we can use loops in various ways.

Types of Loops :

i) for loop :

Syntax :

```
for (initialization; condition; update)
{
        // body

}
```

Q. Print numbers from 1 to 5.

```
for(int num = 1; num ≤ 5; num++){
        System.out.println(num);

}
```

```
1
2
3
4
5
```

num=1,
1≤5 ⟹ goes inside
1++ ⟹ 2 → num
num=2,
2≤5 ⟹ goes inside
2++ ⟹ 3 → num
⋮
soon until 6 ≰ 5

// here in above example, we have initialized a int variable with 1 so now num = 1

then the condition is num ≤ 5 it means until when num is either less or equal to 5, the loop will work and we can enter inside loop. after the code inside run completely then the variable num will update.

We gave num++, it means everytime num will get increment by 1. num++ == num = num+1

After the loops run for 5 times, num get updated to 6 but as 6 ≰ 5 so this time we can't enter the loop and thus loop get run completely.

Conclusion —

first we initialize some or a variable then check condition, if true then we enter ~~inside~~ the loop otherwise we don't and move on from the for loop.

After we enter the loop, code runs and then after completion, the update occurs then again the condition is checked and soon.

This thing continues until the condition is true, as condition become false, we come out of for loop and move on to next line of code.

ii) While loop :

Syntax :

```
initialization;
while (condition){
        // body
        // update
    }
```

→ eg— To print from 1 to 5

```
int num = 1;
while (num < = 5){
        System.out.println(num);
        num++;
}
```

```
1
2
3
4
5
```

<u>Note</u>: Both can be used as our wish but a tip is we can use while() when we don't know how many times the loop will run ~~because at that time~~ for() cannot be used.

iii) <u>do while:</u>

syntax:—

```
        do {
            // body
        } while(condition);
```

// here we first execute body inside do{} and then check the condition inside while().

eg - print from 1 to 5.

```
int num = 1;
do {
        System.out.println(num);
} while (num <= 5);
```

```
1
2
3
4
5
```

Difference in while and dowhile :

⟹ while loop may or may not run depending on the condition

⟹ But do while loop will run atleast once in any condition.

because in while(), condition is checked first and then the code is executed

but in dowhile(), condition is checked at the last so in case, at first try, condition is false then also the code will be run once till the condition checking

Q. Print max of three numbers.

there can be various ways to solve this.

Q. Print if the first letter of string is in upper/lower case

// note :

```
string word = "hello";
System.out.println(word.charAt(0));
```

this means we get char at index 0 of variable "word" ⇒ which is h
charAt() returns a char value.

and trim() removes the space from the beginning and end of the variable (if any)

eg —

```
   hey
 hey
```

// extra space before "hey" got removed by trim()

// code :

```
Scanner in = new Scanner(System.in);
char ch = in.next().trim().charAt(0);

if (ch >= 'a' && ch <= 'z'){
        System.out.println("Lower Case");
} else {

        System.out.println(" Capital/Upper");

}
```

// && is bitwise and which checks both the condition mentioned inside if are true or not, if true → prints the code otherwise goes to else part.

// here due to auto promotion of type, chars get converted to int type and ASCII code are used to replace the char values and then conditions are checked.

Fibonacci Numbers / Series:

Fibonacci Series consists of

index — 0 1 2 3 4 5 6
series — 0 1 1 2 3 5 8 . . .

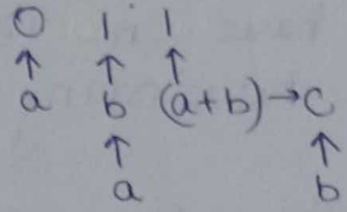
where first two number will be fixed and a current number will be summation of previous two number.

Q. Find the nth fibonacci number.

a = 0
b = 1

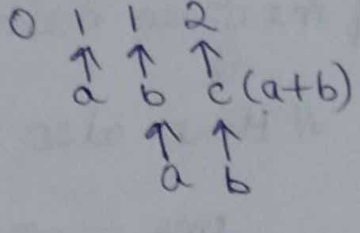
in for(), i = 2, n = 7 then
as i <= 7, so

```
c <- a + b
    a = b
    b = c
then i++;
```



as i = 3 and i <= 7, so

```
c <- a+b
    a = b
    b = c
then i++;
```



as i = 4 and i <= 7, so

⋮

as i = 8 and $i \not\le 7$, so execution of for is complete.

now we will print the "b".

// code :

```
    Scanner inp = new Scanner(System.in);
    int n = in. nextInt();
    int a = 0;
    int b = 1;
    int count = 2;
    int c;
    while (count ≤ n){
                c = a + b;
                a = b;
                b = c;
                count ++;
    }

    System.out. println (b);
```

Q.
we have n = 138575 7879, and we need to check how many times 7 occurs.

→ In such cases, we have to check each digit of the given number.

// Here also we may have different ways,

one ⟶ we can convert the number to string and then by iterating each index we can check

second ⟶ a fact is when we do modulo of any number by 10, we get last digit of number in every case,

eg − n = 1389
→ n%10 → last digit

$$\begin{array}{r} 138 \\ 10 \overline{)1389} \\ 1380 \\ \hline 9 \end{array}$$

9 ← last digit of n.

eg-

n = 13839, check occurrence of 3.

```
while(n>0){
        r = n % 10;   // gives last digit
        if (r == 3) {
            count ++; }
        n = n/10;  // to remove last digit from
                   // given number n
}
```

n = 13839, so n > 0 :

$r = 13839 \% 10 = 9$

if $(r == 3) \Rightarrow$ no

$n = n/10 = 1383$

n = 1383, so n > 0 :

$r = 1383 \% 10 = 3$

if $(r == 3) \Rightarrow$ true :

count $\leftarrow 0+1$

$n = n/10 = 138$

n = 138, so n > 0 :

$r = 138 \% 10 = 8$

if $(r == 3) \Rightarrow$ no

$n = n/10 = 13$

n = 13, so n > 0 :

$r = 13 \% 10 = 3$

if $(r == 3) \Rightarrow$ true :

count $\leftarrow 1+1$

$n = n/10 = 1$

n = 1, so n > 0 :

$r = 1 \% 10 = 1$

if $(r == 3) \Rightarrow$ no

$n = n/10 = 0$

n = 0, so $n > 0 \Rightarrow$ no

// code :

```
int n = 45536;
int count = 0;

while (n > 0):
        int rem = n % 10;
        if (rem == 5){
            count++;
        }
        n = n/10;
}

System.out.println(count);
```

Q. If we have to reverse a number.

// we know, for such case, we'll take one digit each time by doing % and / as done in last example and each time we get a number by doing modulo, it have to be inserted in the new number we'll create one digit after an-other.

we know, 79 means 70 + 9

$$70+9 \Downarrow 7\times 10+9$$

similarly, 102 means

$10\times 10 + 2$

the same logic will be used to insert a digit in new variable of number.

eg - n = 129   n' = reverse of n
n' = 0;
n = 129, n > 0 :
r = 129 % 10 = 9 ;

$n' = n' \times 10 + r ; \Leftarrow 0 \times 10 + 9 = 9$
$n = n/10 ; \Leftarrow 129/10 = 12$

$n = 12, n > 0$:
$r = 12 \% 10 = 2$;
$n' = n' \times 10 + r \Leftarrow 9 \times 10 + 2 = 92$
$n = n/10 \Leftarrow 12/10 = 1$

$n = 1, n > 0$:
$r = 1 \% 10 = 1$
$n' = n' \times 10 + r \Leftarrow 92 \times 10 + 1 = 921$
$n = n/10 \Leftarrow 1/10 = 0$

$n = 0, n \not> 0 : \Rightarrow$ no.

so, now $n' = 921$ (reverse of n).

```
// code :
      int num = 28479;
      int ans = 0;

      while (num >0) {
      int r = num % 10; // to access last digit
      num = num / 10; // to remove last digit
      ans = ans * 10 + r ; // to update ans
      }
      System.out.println(ans);
```

```
97482
```

Calculator Program:

// code:

```
Scanner in = new Scanner(System.in);
int ans = 0;
while (true){
 char op = in.next().trim().charAt(0);
 if (op == '+' || op == '-' || op =='*'
 || op == '/' || op == '%'){
      int num1 = in.nextInt();
      int num2 = in.nextInt();

      if (op == '+'){
          ans = num1 + num2;
      }
      if (op == '-'){
           ans = num1 - num2;
      }
      if (op == '*'){
           ans = num1 * num2;
      }
      if (op == '/'){
            ans = num1 / num2;
      }
      if (op == '%'){
             ans = num1 % num2;
      }
 } else if (op == 'x' || op == 'X'){
        break;
 }else { // for any char other than 7 char
        //mentioned above.
        System.out.println ("Invalid");
      } System.out.println (ans);
   }
```

// here,

```
char op = in.next().trim().charAt(0);
```

since we don't have any specific next() to intake a char data but we have .next() which intakes a string so we do that and after triming and converting it to char data by using charAt(0), we assign it to a variable of char data type.