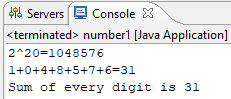
# Number 1

2^15 = 32768 and the sum of its digits is 3 + 2 + 7 + 6 + 8 = 26. What is the sum of the digits of the number 2^20?

Answer :



There’s 2 steps to get the answer

Step one : calculate 2 raised to the power of 20.

Step two : calculate the sum of every digit from step one’s result.

**public** **static** **void** process(**int** a, **int** b) {

//exponent process

String pow\_result = *exponent*(a, b);

System.*out*.println(a + "^" + b + "=" + pow\_result);

//sum every digit process

**int** sum\_result = *sumOfDigit*(pow\_result);

System.*out*.println("Sum of every digit is " + sum\_result);

}

There’s a code for step one :

**public** **static** String exponent(**int** a, **int** b) {

**int** a\_pow\_b = (**int**)Math.*pow*(a, b);

**return** Integer.*toString*(a\_pow\_b);

}

Then, I changed the type value to string because I need to get every digit of the result.

And from step two, the logic is

**public** **static** **int** sumOfDigit(String pow\_result) {

**int** sum = 0;

**for**(**int** i=0; i < pow\_result.length(); i++)

{

sum += (pow\_result.charAt(i) - '0');

}

**return** sum;

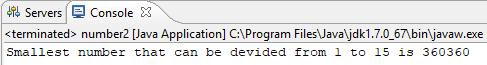
} :

From the result of step one, We get each digits with charAt() command. And that digit is reduced by ‘0’ to return the value into integer. After we get every digits of exponential result, we only need to add the value every loop process.

# Number 2

2520 is the smallest number that can be divided by each of the numbers from 1 to 10 without any remainder. What is the smallest positive number that is evenly divisible by all of the numbers from 1 to 15?

Answer :



I am using loop method to get the smallest number that can be divided by more than 2 numbers. First, we will calculate the smallest number that can be divided by 1 and 2. Then, the result from that will be calculated with 3, and so on until 15.

Lcm(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15) is equal to :

lcm(lcm(lcm(lcm(lcm(lcm(lcm(lcm(lcm(lcm(lcm(lcm(lcm(lcm(1,2),3),4),5),6),7),8),9),10),11),12),13),14),15)

**private** **static** **void** process(**int** n)

{

**int** value = 1;

**for**(**int** i = 1; i < n; i++)

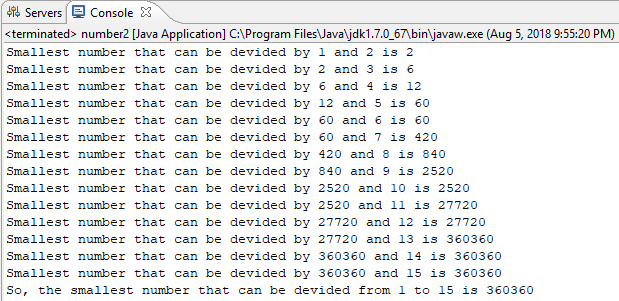
{

value = *lcm*(value, i+1);

}

}

­So, the background process will be like this :



The simple way to get the number that can be divided by two numbers is to multiply that two numbers. Example, we have a and b as natural numbers. So, a\*b can be divided by a and b. but, a\*b not necessarily the smallest number that can be divided by a and b. We must divide a\*b with the largest number that divide a and b.

The code to get smallest number that can be divided by 2 number :

**private** **static** **int** lcm(**int** a, **int** b)

{

**int** lcm\_result = (a \* b) / *gcd*(a, b);

**return** lcm\_result;

}

To get the largest number that divide a and b, I’m using the code bellow :

**private** **static** **int** gcd(**int** a, **int** b)

{

**while** (b != 0)

{

**int** temp = b;

b = a % b;

a = temp;

}

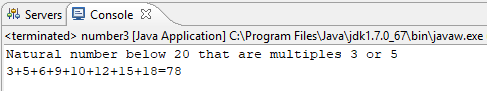
**return** a;

}

# Number 3

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 20

Answer :



To get the answer, I’m using loop command from 1 to n-1 (for this case, n=20).

In the looping process, the system will check whether the number can be divided by 3 or 5 or not. I'm using modulo concept to decide which number that can be divided or not.

If the number can be divided by 3 or 5 (means the modulo is 0), the system will add that number to get the sum.

**public** **static** **void** process(**int** n) {

**int** sum = 0;

**for**(**int** i = 1; i < n; i++)

{

**if**(i%3==0 || i%5==0)

{

sum += i;

}

}

System.*out*.println(sum);

}