Question) **Find whether a given number is EVEN or ODD.**

The above given question is simple. We can easily find whether a number is even or odd using ‘%’ operator.

What is even number?

Even numbers are integer of form 2k where k is integer. Following are even integers.

….,-4, -2, 0, 2, 4, 6, 8,..

What is odd number?

Even numbers are integer of form (2k+1) where k is integer. Following are even integers.

….,-3, -1, 1, 3, 5,..

So we know that number % 2 = 0 then it is even else it is odd.

**public** **class** EvenOdd {

**public** **static** **void** main(String[] args) {

**for** (**int** i = -5; i < 5; i++) {

**boolean** isEven = EvenOdd.*isEven*(i);

**if**(isEven){

System.***out***.println(i+" is Even Number");

}**else**{

System.***out***.println(i+" is Odd Number");

}

}

}

**public** **static** **boolean** isEven(**int** a) {

/\*\*

\* Check whether number is even or odd by using '%'

\* '%' is modules operator that will provide us remainder

\* If remainder is 0 then even

\* else number is odd

\* \*/

**if** ((a % 2) == 0) {

**return** **true**;

} **else** {

**return** **false**;

}

}

}

Ouput:

-5 is Odd Number

-4 is Even Number

-3 is Odd Number

-2 is Even Number

-1 is Odd Number

0 is Even Number

1 is Odd Number

2 is Even Number

3 is Odd Number

4 is Even Number

In the above program we check for the remainder of the number. We exploit the property that even number is divisible by 2 and remainder will be 0(zero). So we use ‘%’ to get remainder. If the remainder is 0(zero) then we say that number is even else it is odd.

Now ‘%’ is costly operation as it will divide the number constantly with the divisor in our case it is 2 and return output 0(zero) or 1.

We can use the bit operation as it will defiantly work faster than continuous division of modules (%).

For bit operations we will now see binary numbers.

|  |  |
| --- | --- |
| Decimal | Binary |
| **0** | **0000** |
| 1 | 0001 |
| **2** | **0010** |
| 3 | 0011 |
| **4** | **0100** |
| 5 | 0101 |

In table even numbers are in bold. There is a pattern the number that is even does not have last bit 1. For additional information on binary number click [here](http://en.wikipedia.org/wiki/Binary_number).

We can do **bit operation** **& with 1**. **If result is 0(zero) then number is even else odd.**

**public** **static** **boolean** isEven(**int** a) {

/\*\*

\* Check whether number is even or odd by using '&'

\* If result is 0 then even

\* else number is odd

\* \*/

**if** ((a & 1) == 0) {

**return** **true**;

} **else** {

**return** **false**;

}

}

Output will be same as before.

Bit operations example ‘& with 1’

Decimal 2 -> 0010

Decimal 1 -> 0001

Output 0000

Decimal 3 -> 0011

Decimal 1 -> 0001

Output 0001