**The fusc function -- Part 1**

7588% of 12060 of671[ecolban](http://www.codewars.com/users/ecolban)

Java

* [TRAIN AGAIN](http://www.codewars.com/kata/the-fusc-function-part-1/train/java)
* [NEXT KATA](http://www.codewars.com/trainer/java)

Details

[Solutions](http://www.codewars.com/kata/the-fusc-function-part-1/solutions/java)

[Discourse (16)](http://www.codewars.com/kata/the-fusc-function-part-1/discuss/java)

* Add to Collection
* |
* Share this kata:

The fusc function is defined recursively as follows:

fusc(0) = 0

fusc(1) = 1

fusc(2n) = fusc(n)

fusc(2n + 1) = fusc(n) + fusc(n + 1)

Your job is to produce the code for the fusc function. In this kata, your function will be tested with small values of n, so you should not need to be concerned about stack overflow or timeouts.

When done, move on to [Part 2](http://www.codewars.com/kata/the-fusc-function-part-2" \t "_blank).

<http://www.codewars.com/kata/the-fusc-function-part-1/java>

* **public class Kata {**
* **public static int fusc(int n) {**
* **if(n == 0 || n == 1)**
* **return n;**
* **int t = n % 2;**
* **return t == 0 ? fusc(n / 2) : fusc(n / 2) + fusc(n / 2 + 1);**
* **}**
* **}**
  + - Best Practices3
    - Clever0
  + 0
  + [Fork](http://www.codewars.com/kumite/new?group_id=59433b79bad7d516f900051e&review_id=5943326b926992ddf30001f9)
  + [Link](http://www.codewars.com/kata/reviews/5943326b926992ddf30001f9/groups/59433b79bad7d516f900051e)
* [dinglemouse](http://www.codewars.com/users/dinglemouse)
* **public class Kata {**
* **public static int fusc(int n) {**
* **return n<2 ? n : fusc(n/2) + (n%2==0 ? 0 : fusc(n/2+1));**
* **}**

**}**

* + - Best Practices1
    - Clever3
  + 0
  + [Fork](http://www.codewars.com/kumite/new?group_id=5943ca2027a78a73c70001f5&review_id=5943326b926992ddf30001f9)
  + [Link](http://www.codewars.com/kata/reviews/5943326b926992ddf30001f9/groups/5943ca2027a78a73c70001f5)
* [shadowmanos](http://www.codewars.com/users/shadowmanos)
* **class Kata {**
* **static int fusc(int n) {**
* **if (n >= 2) {**
* **return n % 2 == 0 ?**
* **fusc(n / 2) :**
* **fusc(n / 2) + fusc(n / 2 + 1);**
* **}**
* **return n;**
* **}**

**}**