Rounding Up To The Nearest Hundred

Asked 11 years, 4 months ago Modified 3 years, 3 months ago Viewed 58k times



35





I came to a part in my java program where I need to round up to the nearest hundred and thought that there was probably some way to do it but I guess not. So I searched the net for examples or any answers and I've yet to find any since all examples appear to be to the nearest hundred. I just want to do this and round UP. Maybe there's some simple solution that I'm overlooking. I have tried Math.ceil and other functions but have not found an answer as of yet. If anyone could help me with this issue I would greatly appreciate it.

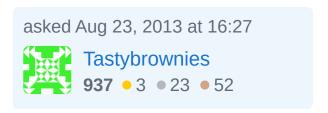
If my number is 203, I want the result rounded to be 300. You get the point.

- 1.801->900
- 2.99->100
- 3. 14->100
- 4. 452->500

java math rounding

Share
Improve this question
Follow





11 Answers

Sorted by:

Highest score (default)





68

Take advantage of integer division, which truncates the decimal portion of the quotient. To make it look like it's rounding up, add 99 first.



```
int rounded = ((num + 99) / 100 ) * 100;
```



Examples:



```
801: ((801 + 99) / 100) * 100 \rightarrow 900 / 100 * 100 \rightarrow 9 *

99: ((99 + 99) / 100) * 100 \rightarrow 198 / 100 * 100 \rightarrow 1 * 1

14: ((14 + 99) / 100) * 100 \rightarrow 113 / 100 * 100 \rightarrow 1 * 1

452: ((452 + 99) / 100) * 100 \rightarrow 551 / 100 * 100 \rightarrow 5 *

203: ((203 + 99) / 100) * 100 \rightarrow 302 / 100 * 100 \rightarrow 3 *

200: ((200 + 99) / 100) * 100 \rightarrow 299 / 100 * 100 \rightarrow 2 *
```

Relevant <u>Java Language Specification quote</u>, <u>Section</u> <u>15.17.2</u>:

Integer division rounds toward 0. That is, the quotient produced for operands n and d that are integers after binary numeric promotion (§5.6.2)

is an integer value q whose magnitude is as large as possible while satisfying $|d \cdot q| \le |n|$.

Share Improve this answer Follow

edited Jul 15, 2014 at 3:50



answered Aug 23, 2013 at 16:29



1 Wow, I never thought of taking advantage of truncation like that. This answer is pretty awesome. Thank you very much for teaching me something! – Tastybrownies Aug 23, 2013 at 16:34

*Side note --- if you are working with float point values, rather than casting an int , most languages support a floor function in some way or another. – Albert Renshaw Jul 15, 2014 at 3:52

@DaSh Yes it works. Rounding 0 up to the nearest hundred is 0, because 0 is the nearest multiple of 100, and ((0 + 99) / 100) * 100 -> 99 / 100 * 100 -> 0 * 100 = 0. - rgettman Dec 1, 2014 at 17:16

@rgettman Excellent! Simple and elegant. I was trying to make some kind of equation for this but failed. – kirtan403 Dec 21, 2016 at 12:15



Here is an algorithm which I belive works for any "multiple of" case. Let me know what you think.





```
int round (int number,int multiple){
  int result = multiple;

  //If not already multiple of given number
  if (number % multiple != 0){
    int division = (number / multiple)+1;
    result = division * multiple;
  }
  return result;
}
```

Share Improve this answer Follow

answered Aug 24, 2013 at 7:10
O.C.
6,819 • 1 • 27 • 27

1 Thanks worked for me. Only alteration I felt is first line should be: int result = number; − vanval Apr 20, 2017 at 20:06 ✓



Try this:

9

(int) (Math.ceil(number/100.0))*100



Share Improve this answer Follow

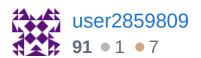


Claus Wilke 17.7k • 8 • 59 • 107

(1)

answered Nov 27, 2017 at 23:58

edited Nov 28, 2017 at 0:59





5







```
int roundUpNumberByUsingMultipleValue(double number, i
        int result = multiple;
        if (number % multiple == 0) {
            return (int) number;
        }
        // If not already multiple of given number
        if (number % multiple != 0) {
            int division = (int) ((number / multiple)
            result = division * multiple;
        }
        return result;
    }
Example:
System.out.println("value 1 = " + round(100.125, 100));
System.out.println("value 2 =" + round(163,50));
System.out.println("value 3 =" + round(200, 100));
System.out.println("value 4 =" + round(235.33333333,10
System.out.println("value 5 =" + round(0,100));
OutPut:
value 1 = 200
value 2 = 200
value 3 = 200
value 4 = 300
value 5 = 0
```



- 1 Posting just a piece of code does not help a lot, you should consider adding some explanation to you answers.
 - modusCell Jul 19, 2014 at 13:57



```
long i = 2147483648L;
if(i % 100 != 0) {
   long roundedI = (100 - (i \% 100)) + i;
}
```



Example:





```
649: (100 - (649 \% 100)) + 649 -> <math>(100 - 49) + 649) ->
985: (100 - (985 % 100)) + 985 -> (100 - 85) + 985) ->
```

Long datatype is used to make sure the limitation of integer range should not cause any problem for larger values. For ex, this might be very important in case of an amount value (banking domain).

Share Improve this answer Follow

edited Oct 1, 2016 at 21:09

answered Oct 1, 2016 at 20:46



Mansingh Shitole **151** • 1 • 6



One other way is to use BigDecimal





private static double round(double number, int precisi roundingMode) { BigDecimal bd = null; try { bd = BigDecimal.valueOf(number);

```
} catch (NumberFormatException e) {
    // input is probably a NaN or infinity
    return number;
}
bd = bd.setScale(precision, roundingMode);
return bd.doubleValue();
}

round(102.23,0,RoundingMode.UP) = 103
round(102.23,1,RoundingMode.UP) = 102.3
round(102.23,2,RoundingMode.UP) = 102.24
round(102.23,-1,RoundingMode.UP) = 110
round(102.23,-2,RoundingMode.UP) = 200
round(102.23,-3,RoundingMode.UP) = 1000
```

Share Improve this answer Follow

answered Apr 20, 2017 at 20:25

vanval
1,027 • 1 • 11 • 19



A simple implementation of rgettman trunaction:

```
3
```





```
public class Main {

   private static int roundUp(int src) {
      int len = String.valueOf(src).length() - 1;
      if (len == 0) len = 1;
      int d = (int) Math.pow((double) 10, (double) l
      return (src + (d - 1)) / d * d;
   }

   public static void main(String[] args) {
      System.out.println("roundUp(56007) = " + round
      System.out.println("roundUp(4511) = " + roundU
      System.out.println("roundUp(1000) = " + roundU
      System.out.println("roundUp(867) = " + roundUp(
      System.out.println("roundUp(5) = " + roundUp(5)
      System.out.println("roundUp(5) = " + roundUp(5)
      System.out.println("roundUp(0) = " + roundUp(0)
```

```
}
```

Output:

```
roundUp(56007) = 60000
roundUp(4511) = 5000
roundUp(1000) = 1000
roundUp(867) = 900
roundUp(17) = 20
roundUp(5) = 10
roundUp(0) = 0
```

Share Improve this answer Follow



answered Oct 24, 2016 at 7:10



Perfect solution for my use case Thanks a lot – Rupesh Patil Apr 18, 2023 at 10:23



I don't have enought reputation to add a comment to O.C.'s answer but I think it should be:

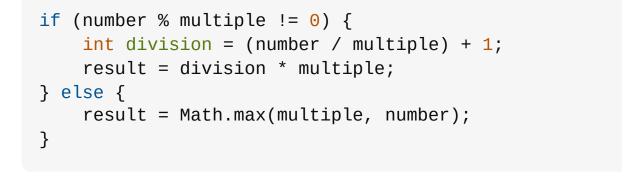
1











with the else so that, for example round(9, 3) = 9, otherwise it would be round(9, 3) = 3

Share Improve this answer Follow

edited Sep 2, 2021 at 7:17

Nadeem Iqbal
2,374 • 1 • 30 • 45

answered Sep 21, 2016 at 9:33





The below code works for me to round an integer to the next 10 or 100 or 500 or 1000 etc.

0





```
public class MyClass {
   public static void main(String args[]) {
      int actualValue = 34199;
      int nextRoundedValue = 500 // change this base
requirment ex: 10,100,500,...
   int roundedUpValue = actualValue;

   //Rounding to next 500
   if(actualValue%nextRoundedValue != 0)
      roundedUpValue =
(((actualValue/nextRoundedValue)) * nextRoundedValue)
      System.out.println(roundedUpValue);
   }
}
```

Share Improve this answer Follow

edited Feb 28, 2018 at 6:15



Andreas **5.579** • 10 • 47 • 55

answered Feb 28, 2018 at 5:55





This worked perfectly for me:









```
var round100 = function(n) {
    if (n < 50) {
       var low = n - (n % 100);
       return Math.round(low);
    }

    return Math.round(n/100) * 100;
}</pre>
```

You can assign your var (variables) to anything.

Share Improve this answer Follow

answered Mar 21, 2019 at 2:22





-1



```
int value = 0;
for (int i = 100; i <= Math.round(total); i = i + 100
    if (i < Math.round(total)) {
       value = i;
     }
}</pre>
```

It will run until total is == to i and i will be increased every time

Share Improve this answer

answered Feb 9, 2021 at 7:06

Bhavik Maradiya

1

Follow

