Test these out in your project.

|  |
| --- |
| /\* Write a method largestAbsVal that accepts three integers as parameters and |
|  | \* returns the largest of their three absolute values. |
|  | \*/ |
|  | public int largestAbsVal(int a, int b, int c) { |
|  | return Math.max(Math.abs(a), Math.max(Math.abs(b), Math.abs(c))); |
|  | } |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | /\* Write a method called distance that accepts four integer coordinates x1, y1, | |  | \* x2, and y2 as parameters and computes the distance between points (x1, y1) | |  | \* and (x2, y2) on the Cartesian plane. | |  | \*/ | |  | public double distance(int x1, int y1, int x2, int y2) { | |  | return Math.sqrt((x2 - x1) \* (x2 - x1) + (y2 - y1) \* (y2 - y1)); | |  | } | |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

========================================

|  |
| --- |
| /\* Write a method named lastDigit that returns the last digit of an integer. |
|  | \* It should work for negative numbers as well. |
|  | \*/ |
|  | public int lastDigit(int n) { |
|  | return Math.abs(n % 10); |
|  | } |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | /\* Write a method called printPowersOf2 that accepts a maximum number as an | |  | \* argument and prints each power of 2 from 20 (1) up to that maximum power, | |  | \* inclusive.You may assume that the value passed to printPowersOf2 is 0 or | |  | \* greater. | |  | \*/ | |  | public void printPowersOf2(int n) { | |  | for(int i = 0; i <= n; i++) | |  | System.out.print(((int) Math.pow(2, i)) + " "); | |  | } | |
|  |  |
| |  | | --- | | /\* Write a method called getGrade that accepts an integer representing a | |  | \* student's grade in a course and returns that student's numerical course | |  | \* grade. The grade can be between 0.0 (failing) and 4.0 (perfect). Assume | |  | \* that scores are in the range of 0 to 100. For an added challenge, make your | |  | \* method throw an IllegalArgumentException if the user passes a grade lower | |  | \* than 0 or higher than 100. | |  | \*/ | |  |
|  |  |
| |  | | --- | | /\* Write a method named isAllVowels that returns whether a String consists | |  | \* entirely of vowels (a, e, i, o, or u, case-insensitively). If every | |  | \* character of the String is a vowel, your method should return true. If any | |  | \* character of the String is a non-vowel, your method should return false. | |  | \* Your method should return true if passed the empty string, since it does | |  | \* not contain any non-vowel characters. | |  | \*/ | |  | public boolean isAllVowels(String str) { | |  | String vowels = "aeiouAEIOU"; | |  |  | |  | for(int i = 0; i < str.length(); i++) { | |  | if(vowels.indexOf(str.charAt(i)) == -1) | |  | return false; | |  | } | |  |  | |  | return true; | |  | } | |  |
|  |  |
|  |  |
|  |  |