Tasks for Live Coding exercises during module “Java Fundamentals - programming”

www.sdacademy.pl

Task 1.

Write an application that will read diameter of a circle (variable of type *float*) and calculate perimeter of given circle.
Firstly, assume π = 3.14. Later, use value of π from built-in *Math* class.

Task 2.

Write an application calculating BMI (Body Mass Index) and checking if it’s optimal or not. Your application should read two variables: weight (in kilograms, type *float*) and height (in centimeters, type *int*). BMI should be calculated given following formula:

$$BMI = \frac{weight\,[kg]}{height[m]^2}$$

The optimal BMI range is from 18.5 to 24.9, smaller or larger values are non-optimal values. Your program should write "BMI optimal" or "BMI not optimal", according to the assumptions above.

software **development** academy

www.sdacademy.pl

Task 3.

Write a program for solving a quadratic equation. The program should take three integers (coefficients of the quadratic equation *a, b, c*) and calculate the roots x_1, x_2 of the equation $ax^2 + bx + c = 0$

If delta Δ comes out negative, print "Delta negative" and exit the program.

Formulas you'll need:

$$\Delta = b^2 - 4ac$$

$$x_1 = \frac{-b - \sqrt{\Delta}}{2a}$$

$$x_2 = \frac{-b + \sqrt{\Delta}}{2a}$$

Task 4.

Write an application that takes a positive number from the user (type *int*) and writes all numbers from 1 to the given number, each on the next line, with the following changes:

- in place of numbers divisible by 3, instead of a number the program should print "Fizz"
- in place of numbers divisible by 7, instead of a number the program should write "Buzz"
- if the number is divisible by both 3 and 7, the program should print "Fizz buzz"

software **development** academy

Task 5.

Write an application that takes a positive number from the user (type *int*) and prints all prime numbers greater than 1 and less than the given number.

software **development** academy

Task 6.

Write an application that takes a number *n* from the user (type *int*) and calculates the sum of the harmonic series from 1 to *n*, according to the formula below:

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots + \frac{1}{n}$$

Task 7.

Write an application that will take a positive number from the user (type *int*) and calculate the Fibonacci number at the indicated index. For example, if the number equals 5, your program should print the fifth Fibonacci number. In Fibonacci sequence, each number is the sum of the two preceding ones. For example, the first few Fibonacci numbers are:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377…

Task 8.

Write an application that implements a simple calculator. The application should:

a. read first number (type *float*)

b. read one of following symbols: + - / *

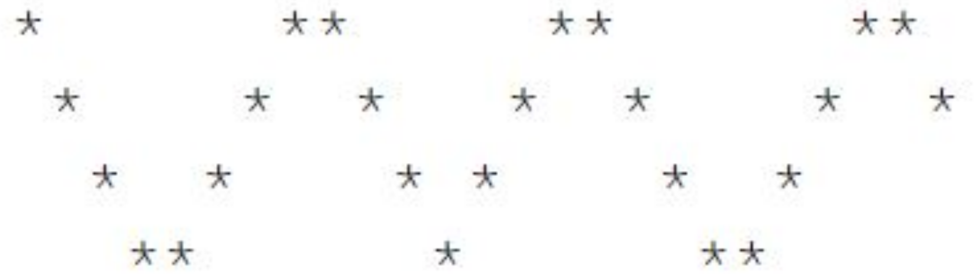
c. read second number (type *float*)

d. return a result of given mathematical operation

If the user provides a symbol other than supported, the application should print "Invalid symbol". If the entered action cannot be implemented (i.e. it is inconsistent with the principles of mathematics), the application should print "Cannot calculate".

Task 9.

Write an application that will take a positive number from the user (type *int*) and draw a wave with a given length and height of 4 lines, according to the following example (fill empty spaces with spaces):



```
*      **     **      **
 *    *  *   *  *    *  *
  *  *    * *    *  *
   **      *      **
```

Task 10.

Write an application that gets one positive number (type *int*) from the user and calculates a sum of digits of the given number. Hint: to make some operations on every single digit of the number (digit by digit), you can calculate the remainder of dividing the number by 10 (to get the value of the last digit) and divide the number by 10 (to "move" to the next digit).

Task 11.

Write an application that will read texts (variables of the *String* type) until the user gives the text "Enough!" and then writes the longest of the given texts (not including the text "Enough!"). If the user does not provide any text, write "No text provided".

Task 12.

Write an application that reads a text from the user (type *String*) and counts a percentage of occurrences of a space character.

Task 13.

Write an application that "stutters", that is, reads the user's text (type *String*), and prints the given text, in which each word is printed twice.

For example, for the input: "This is my test" the application should print "This This is is my my test test".

Task 14.

Write an application that reads two lowercase letters of the Latin alphabet (type char) and calculates how many characters is there in the alphabet between given letters. Hint - use the ASCII code table and treat the characters as int numbers.

Task 15.

Write an application that reads from the user 10 arbitrarily large numbers (variables of type *int*) and prints those that occurred at least twice.

Task 16.

Write an application that takes 10 numbers from the user (type *int*) and write the length of the longest such subsequence of these numbers, which is increasing.

For example, for the numbers: "1, 3, 8, 4, **2, 5, 6, 11, 13**, 7" the program should write "5" as the length of the longest increasing subsequence (underlined in the example).

Task 17.

Write an application that will read from the user the date of your next SDA classes and calculate how many days are left to them.

Hint: read the date as *String* and parse it to *LocalDate*. Get the current date using *LocalDate.now()* method.

Task 18.

Write an application that reads a text from the user (type *String*) and checks whether the user sneezed, i.e. whether the text equals ‘achooo!’ with one or more letter "o" at the end of the expression (so both 'acho!' and 'achooooooo!’ are correct). Hint: use a regular expression with the appropriate quantifier.

Task 19.

Write an application that consists of few classes:

a. *Author* class, representing an author – poem writer, which consists of fields *surname* and *nationality* (both of type *String*)

b. *Poem* class, representing poem, which consists of fields *creator* (type *Author*) and *stropheNumbers* (type *int* – numbers of strophes in poem)

c. *Main* class, with *main* method, inside which you will:

   i. Create three instances of *Poem* class, fill them with data (using constructor and/or *setters*) and store them in array

   ii. Write a surname of an author, that wrote a longest poem (let your application calculate it!)

Task 20.

Write an application that will play "too much, too little" game with you. At the beginning the application should randomly choose a number from 0 to 100 (hint: use the *Random.nextInt()* method) and wait for the user to enter a number. If the user gives a number greater than the number chosen by the computer, your application should print "too much" and wait for the next number. If the user gives a smaller number, the application should print "not enough" and wait for the next number in the same way. If the user provides the exact value, the application should print the word "Congratulations!" and finish.