# Hands-on Experiment for Lecture 7 Method: Worksheet

Section\_\_\_\_\_\_\_1\_\_\_\_\_\_\_ Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_22/9/2020\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student ID \_\_\_\_\_\_\_\_\_\_\_6338110221\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Name\_\_\_Nonthapat Kaewamporn\_\_\_\_\_\_\_\_\_\_\_\_\_

## Part A: Basic Method

* You are given the following equation:

Write a program (TwoTermsMethod.java) that takes (from keyboard) the value of a, b, n, x, r (**all variables are of type int and all calculation must result in double**) then computes **y**. Finally, you must call a method to print all inputs and output (y).

A picture containing object, clock

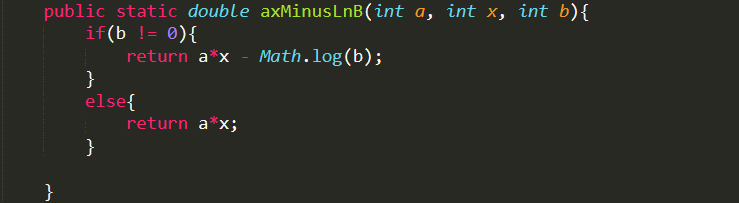
Description automatically generated

Hint: We can write this program by:

* Write a method that calculates . Also, when b is 0, the result is just
* Write a method that calculates , using the method we defined earlier.
* Write a method that calculates that calls both previous methods
* Write a method to print all inputs (a, b, n, x, r) and output (y), where *y* must be displayed in two decimal points.

The method that calculates has the following header:  
**public** **static** **double** axMinusLnB (**int** a, int x, int b)

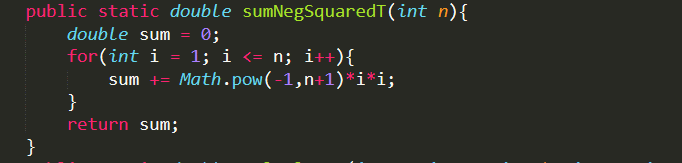
List the source code of the method here:



The method that calculates has the following header:

**public** **static** **double** sumNegSquaredI(**int n**)

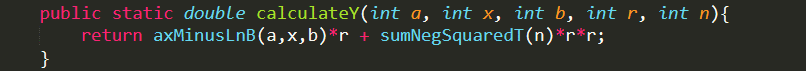
List the source code of the method here:



The method that calculates the value of y has the following header:

**public** **static** **double** calculateY(**int** a, int x, int b, int r, int n)

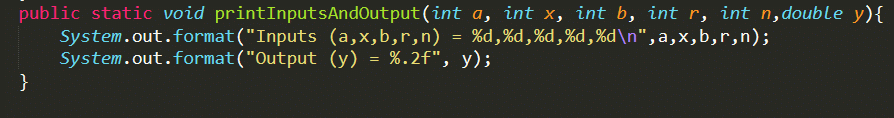
List the source code of the method here:



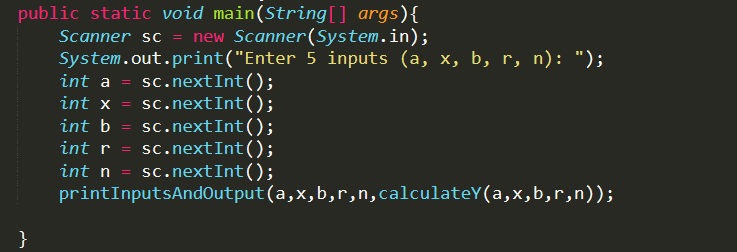
The method that prints all inputs and output the following header:

**public** **static** **void** printInputsAndOutput(**int** a, int x, int b, int r, int n, double y)

List the source code of the method here:



List the source code of your **main** method below:



Write the result of your program in the table below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | x | b | r | n | y |
| 1 | 2 | 3 | 4 | 5 | 883.61 |
| 1 | 4 | 2 | 5 | 6 | -2258.47 |
| 2 | 5 | 1 | 3 | 7 | 1290.00 |
| 3 | 6 | 0 | 7 | 5 | 2821.00 |

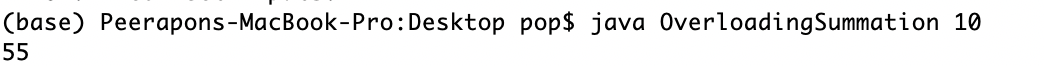
## Part B: Overloading Method

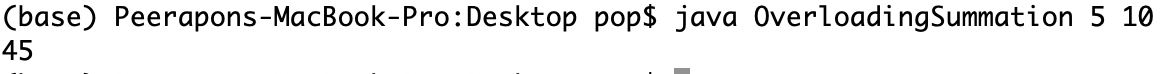
Create a java program (OverloadingSum.java) to create three overloading sum methods as follows:

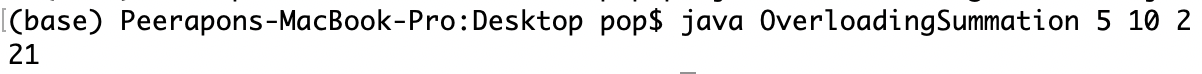
1. int sum(int end); 🡪 return a summation from 1 to *end* with an increment by 1
2. int sum(int start, int end); 🡪 return a summation from *start* to *end* with an increment by 1
3. int sum(int start, int end, int step); 🡪 return a summation from *start* to *end* with an increment by *step*
4. int sum(int start, int end, int step, int bias); 🡪 It calculates similarly to the previous option, and then topup (add) with *bias*.

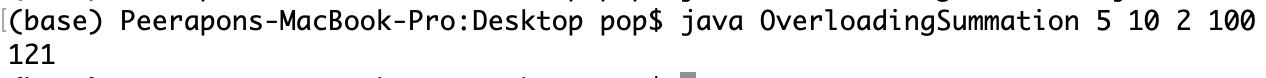
In the main method, the program must check the number of input argument to call the proper summation method. Also, if the number of inputs is incorrect or inputs are NOT integer (NumberFormatException), the program should print “"ERROR: incorrect inputs!"”.

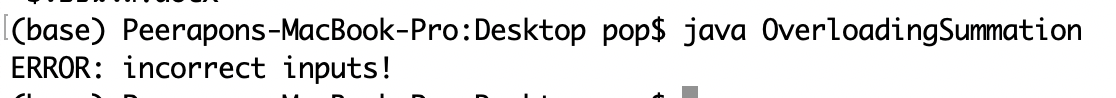
Hint: args.length returns the number of arguments (int) and Integer.parseInt(String) should be helpful. The followings are output examples.

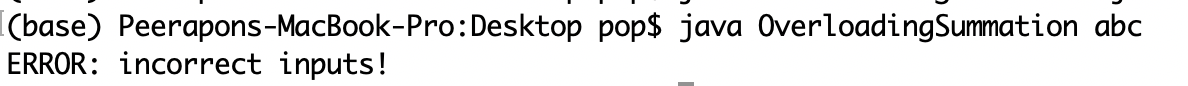












List the source code of the method here:

