Lesson 5. Methods

Exercise 1. Develop a Java class with the following methods:

- A method to calculate the factorial of an integer number.
    - NAME:
        - `factorial`
    - PARAMETERS:
        - Integer number *n >= 0*
    - RETURNS:
        - Integer number *result = n! = 1 · 2 · 3 · … · (n-2) · (n-1) · n*
          E.g.: $0! = 1; 1! = 1; 5! = 120$

- A method to calculate the binomial coefficient of two integer numbers –based on `factorial`.
    - NAME:
        - `bCoefficient`
    - PARAMETERS:
        - Integer number *n >= 0*
        - Positive integer number *m <= n*
    - RETURNS:
        - Integer number *result* = $\binom{n}{m} = \frac{n!}{m!(n-m)!}$
          E.g.: $\binom{7}{3} = 35$

- A `main` method to calculate the binomial coefficient of two integer numbers passed as arguments to the program.

  The main method must print on the screen the binomial coefficient $\binom{n}{m}$, being *n* the first argument and *m* the second argument. The program must check the possible errors in the format (e.g.: wrong number format) and the value (e.g. negative values) of the arguments.

- Modify the previous `main` method to read the values of *n* and *m* from the keyboard. The program must present a choice to let the user continue reading values or end the program.

Exercise 2. Develop a Java method to calculate the b-th power a^b without using the `Math` methods. The method must receive *a* (real) and *b* (positive integer) as parameters and return a double value as a result.

Exercise 3. Based on the methods created in 1 and 2, develop a method named `approximation_e` to calculate an approximation to the number e^x with precision *n* according to the following formula:

$$e^x \approx \sum_{k=0}^{n} \frac{x^k}{k!}$$

For example:

- x = 1, n = 5

$$e^1 = e \approx \sum_{k=0}^{5} \frac{1^k}{k!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} = 1 + 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{24} + \frac{1}{120} = 2,7166$$

- x = 2, n = 4

$$e^2 \approx \sum_{k=0}^{4} \frac{2^k}{k!} = \frac{1}{0!} + \frac{2}{1!} + \frac{2^2}{2!} + \frac{2^3}{3!} + \frac{2^4}{4!} = 1 + 1 + \frac{4}{2} + \frac{8}{6} + \frac{16}{24} = 6$$

The method `approximation_e` receives *n* (integer) and *x* (real) as parameters and returns the approximation to *e* (real) resulting from the calculation.

Exercise 4. Develop a program to calculate various approximations to e^x by using the method `approximation_e` for different values of *n*.

The program must read two values from the keyboard: *x* (real) and *max* (positive integer):

- *x* is the exponent of the approximation e^x
- *max* determines the maximum value of *n* that will be used to calculate the approximation; i.e., the method `approximation_e` will be executed for *n* = {1, 2, …, *max*}

The program must print on the screen the value of n, the value of the approximation for this n, and the value of e^x as obtained with the `Math.exp` method.

For example, for x=2 and max=10, the output must be:

```
Enter x: 2
Enter max: 10

   n    | Approximation e | Math library
     1 |        3,000000 |  7,389056
     2 |        5,000000 |  7,389056
     3 |        6,333333 |  7,389056
     4 |        7,000000 |  7,389056
     5 |        7,266667 |  7,389056
     6 |        7,355556 |  7,389056
     7 |        7,380952 |  7,389056
     8 |        7,387302 |  7,389056
     9 |        7,388713 |  7,389056
    10 |        7,388995 |  7,389056
```

(Note that, due to range and precision of data types, the previous values may be different. The results for large values of x and max may result in values out of the range of the types and be incorrect. Use *printf* to format the output.)

Exercise 5. Develop a class named Matrix to implement the following methods involving matrices –represented as two-dimension arrays.

If the dimensions of the matrices passed as parameters to the methods are not correct to perform the corresponding operation, the method must return the special value `null`.

a. Random initialization

Parameters: Matrix `double [][] m`, `double a`, `double b`

Returns: Nothing

Operation: Initialize `m` with random values in the range [a, b)

b. Read matrix values

Parameters: Matrix `double [][] m`

Returns: Nothing

Operation: Reads `m` values from the keyboard

c. Print matrix

Parameters: Matrix `double [][] m`

Returns: Nothing

Operation: Prints `m` on the screen

d. Find the maximum value of the array

Parameters: Matrix `double [][] m`

Returns: Matrix `double max`

Operation: max value of the array

e. Addition

Parameters: Matrices `double [][] m1, double [][] m2`

Returns: Matrix `double [][] r`

Operation: `r` = `m1` + `m2`

f. Subtraction

Parameters: Matrices `double [][] m1, double [][] m2`

Returns: Matrix `double [][] r`

Operation: `r` = `m1` - `m2`

g. Scalar multiplication

Parameters: Matrices `double [][] m, double x`

Returns: Matrix `double [][] r`

Operation: r = x x m

h. Multiplication

Parameters: Matrices double [][] m1, double [][] m2

Returns: Matrix double [][] r

Operation: r = m1 * m2

i. Transpose

Parameters: Matrix double [][] m

Returns: Matrix double [][] r

Operation: r = transpose(m)

Programming – Grado en Ingeniería Informática

Authors

Of the English version:
Juan Gómez Romero

Based on the work by:
Ángel García Olaya
Manuel Pereira González
Silvia Castro García
Gustavo Fernández-Baillo Cañas
Daniel Pérez Pinillos
Javier Ortiz Laguna
Álvaro Torralba Arias de Reyna

Universidad Carlos III de Madrid