Exercises 07

1. [EX] **CountLetters**
    a. Consider the source file **CountLetters.java**. Compile and run it. See if it works as expected.
    b. Now, try to enter some text with a space (‘ ‘) or a punctuation mark (‘!”) or a digit. Does it produce a correct result?
    c. Enhance the code with exception handling so that any unexpected word (containing non-alphabetic characters – English letters) are basically ignored.

2. [PW] **Factorial**
    a. Consider the source file **Factorial.java**. Compile and run it. See if it works as expected.
    b. What does it produce if a negative value is entered? Do you think it is the correct answer you are getting?
    c. If such number is provided to the **factorial(int)** method, throw an exception of type **IllegalArgumentException**.
    d. Does to compiler impose you to handle it back in main? Why or why not?
    e. Add handler to main and see if anything changes when a negative value is provided as input.

3. [PW] **Invoice class**
    a. Recall that you have implemented an Invoice class.
    b. In setter methods and the constructors you had validation, quantity and price cannot be negative values.
    c. Throw **IllegalArgumentException** and handle it in the main method of the Main class where you create and use Invoice.

4. [PW] **Account class**
    a. Recall that you have implemented an Account class.
    b. In ***withdraw(double amount)*** method there was a validation. Amount cannot be more than the balance. Throw an exception of a new type (define a new Runtime exception: InvalidAmountException)
    c. Throw an instance of it from withdraw method and handle it in the main method of the Main class where you create and use Accout.

5. [EX] **CustomUserGeneration**
    a. Try the last example in the lecture. Make sure you understand the process.
    b. Assume that your user has also some roles associated with it. So the create user takes one parameter the list of roles and use it in the generation of the user.
    c. Make sure you prevent the case when null or empty role list provided as input.

6. [EX] **ParseNumbers**
    a. Consider the source file **ParseNumbers.java**. Compile and run it. See if it works as expected.
    b. Try to provide an input string which contains not only numbers but also, say, letters. Does it produce a correct result?

c. Try to handle it so that all non-number input are ignored.
d. Your program is supposed to produce results even there is no valid input provided.

7. [Bonus] **Stack**
    a. In case you have not heard of Stack ADT read about it.
    b. Consider the source file **Stack.java**. Compile and run it. Try to understand its basic operations.
    c. There are some erroneous cases:
        i. Try to pop() or peek() when the stack is empty.
        ii. Try to push() a new element when the stack is full.
    d. See how you can handle this case? Maybe introduce a new type of exception (or multiple of them).