Introduction to Programming I 2020 - Assignment #3

Deadline: 1/12/2020

You are expected to implement methods that do basic mathematical operations on a given list of numbers (of different types) automatically generated or manually added by the user. In your main method, the user has to be able to choose if he/she wants to insert numbers manually or generate them automatically. The class *MyNumberCreator* has to extend *NumberCreator* abstract class and to implement its methods mentioned below:

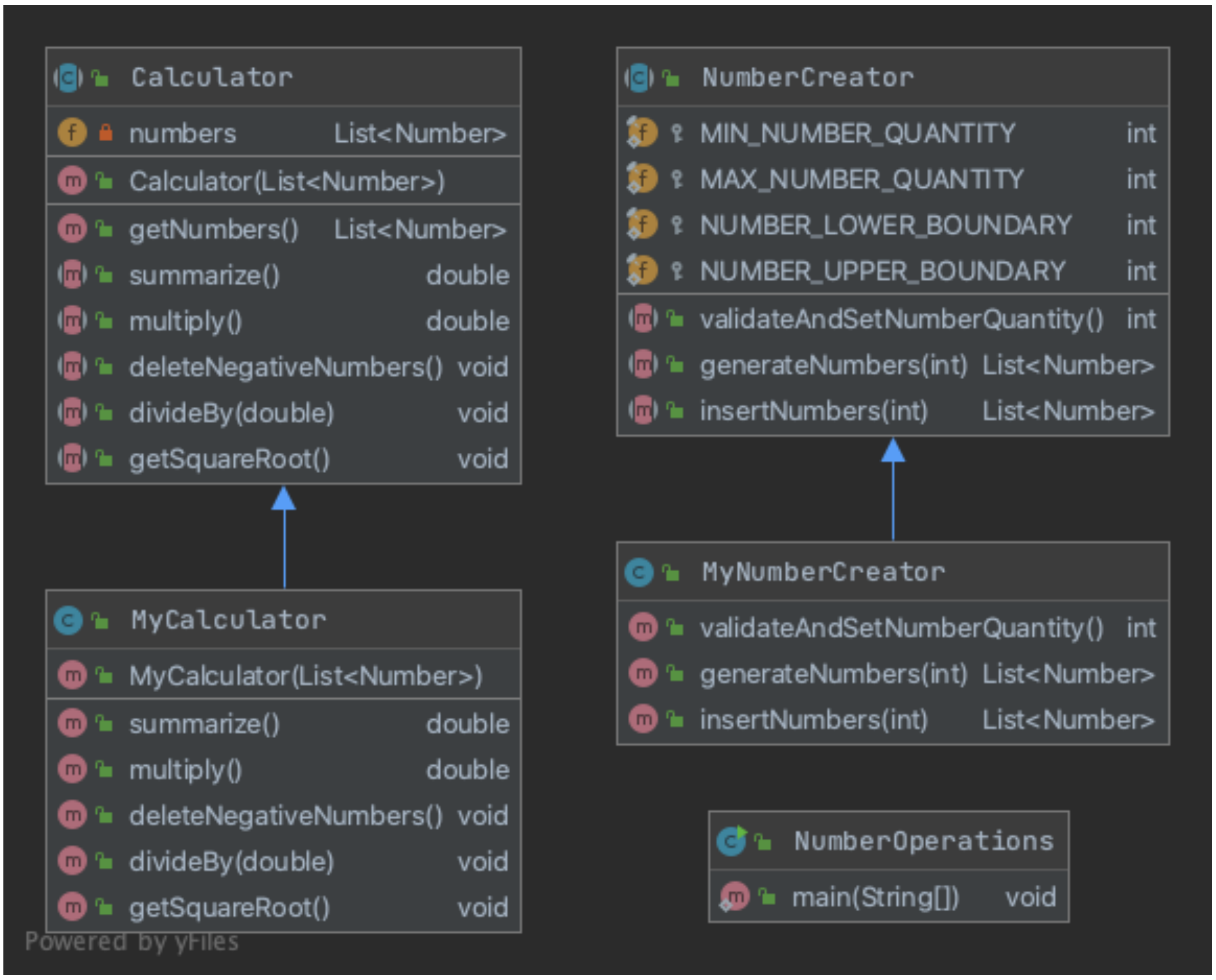
- + *validateAndSetNumberQuantity(): int* allows the user to insert the number of numbers in the list to be processed
- + *generateNumbers(int numberQuantity): List<Number>* allows the generation of a list of random numbers (whose type extends the class *Number*)
- + *insertNumbers(int numberQuantity): List<Number>* allows manual insertion of numbers of random types into the list. The only difference from the previous method is the necessity to type values manually and validate them. Types of numbers should be generated randomly.

The list of subclasses of *Number* in Java includes *Integer*, *Double*, *Byte* and others. All of these types can be included in the list of numbers.

The class *MyCalculator* has to extend *Calculator* abstract class and implement its methods:

- + *summarize(): double* method should allow summarizing all elements of the input list of numbers and should return the double sum of all elements.
  - For example, for list containing [1, -5.5, 26.4693, -59.7, 0.1111] the result should be -37.6196
- + *multiply(): double* should allow multiplying all elements of the input list of numbers and should return the double result of all element multiplication
  - For example, for list containing [1, -5.5, 26.4693, -59.7, 0.1111] the result should be 965.5917261705
- + *deleteNegativeNumbers(): void* should allow deleting all negative numbers from the list and updating the list
  - For example, for list containing [1, -5.5, 26.4693, -59.7, 0.1111] the result should be [1, 26.4693, 0.1111]
- + *divideBy(double divisor): void* should allow dividing each number of the list by the divisor and updating the list values saving the same types
  - For example, for list containing [1, -5.5, 26.4693, -59.7, 0.1111] and divisor = 2.0 the result should be [0, -2.75, 13.23465, -29.85, 0.05555]
- + *getSquareRoot(): void* should allow calculating square root for each element of the array and updating the list values saving the same types (for BigDecimal the precision of Double is considered valid)
  - For example, for the list containing [1, -5.5, 26.4693, -59.7, 0.1111] the result should be [1, ArithmeticException because of negative value

To test your code, the class *NumberOperations* should be created, which contains the *main()* method. However, this class should NOT be submitted. For testing your code, your TA will use his own class. Do not throw exceptions to the main class and do not put any of the logic into the *main()*. Each expected exception should be handled inside your other classes, meaning that incorrect inputs should not lead to the exit from the program. They have to be updated until receiving the valid state, if applicable. Do not rely on this class; it will not be used during testing.


Calculator
numbers List<Number>
Calculator(List<Number>)
getNumbers() List<Number>
summarize() double
multiply() double
deleteNegativeNumbers() void
divideBy(double) void
getSquareRoot() void
NumberCreator
MIN_NUMBER_QUANTITY int
MAX_NUMBER_QUANTITY int
NUMBER_LOWER_BOUNDARY int
NUMBER_UPPER_BOUNDARY int
validateAndSetNumberQuantity() int
generateNumbers(int) List<Number>
insertNumbers(int) List<Number>
MyCalculator
MyCalculator(List<Number>)
summarize() double
multiply() double
deleteNegativeNumbers() void
divideBy(double) void
getSquareRoot() void
MyNumberCreator
validateAndSetNumberQuantity() int
generateNumbers(int) List<Number>
insertNumbers(int) List<Number>
NumberOperations
main(String[]) void
Powered by yFiles

j

1. Software Requirements:

1. Extend the given abstract classes. ALL your additional classes, code, EVERYTHING, should be in the package **numbers.yourname_yoursurname**
2. You are not allowed to change the files in **numbers.do_not_change**, but you can extend/implement and then use the subclasses
3. Your project should be compatible with JDK 1.8
4. You are not allowed to use additional libraries and use frameworks and plugins adding new functionalities to Java, e.g. *Lombok*

2. Assignment Requirements:

1. There should be JavaDoc before each method explaining what it does
    a. See https://www.tutorialspoint.com/java/java_documentation.htm
    b. Good IDEs can generate a template automatically

2. Follow the given Class Diagram
3. The class with the *main()* method should NOT be submitted
4. Submit to Moodle ONLY the zipped (.ZIP) directory **numbers.yourname_yoursurname**
5. Use safe scanning of numbers with throwing and handling exceptions if needed inside your classes. Prefer using exceptions instead of using *if-else* statements if applicable

3. Grading schema (total of 10 points):

- 3 points for code quality (structure, naming, indentation, formatting, no extremely long methods, no magical numbers)
- 2 points for clear comments for java documentation according to the standard mentioned (2.1)
- 3 points for the general impression about the execution of the program and creativity
- 2 points for robustness. For example, replacing your class *MyCalculator* by the same class implemented by one of your colleagues should not break your code
- -5 for failing in implementing the Class Diagram
- -1 for missing (2.4)
- Total grade equals to 0 if your program does not compile
- Total grade equals to 0 if you changed one of the classes/interfaces you were told not to change
- Total grade equals to 0 if caught in the automatic plagiarism check