Assignment 5

Topics: Exception handling, and Design by Contract

NOTE: exercises with prefix * are not mandatory**

1. Read about exceptions at: *http://docs.oracle.com/javase/tutorial/essential/exceptions*

2. Please read and provide at least two A4pages comments on the two papers "*rp_dbc.pdf*" and "*rp_jml.pdf*" that you can find in LearnIT in the compressed archive named "*research_papers.rar*", section "*General resources and material*". Your comments should put the content of the papers in relation to the Java programming language

3. ***Try to get the Java Modeling Language JML which is a behavioral interface that can be used as a design by contract language for Java (see: *http://www.cs.ucf.edu/~leavens/JML/*) working inside NetBeans or Eclipse and explain how this trial went as part of the final document of point 1 above. You can find some help in reading the two additional elective papers "*rp_jml_elective_1.pdf*", and "*rp_jml_elective_2.pdf*", respectively, which are stored in the same compressed archive as for point 2 above.
   NOTE: there exists a plug-in for Eclipse that makes it possible to integrate JML into the IDE. In order to see how to get JML as part of the Eclipse, you can watch the video *JML2 PlugIn.mp4* which you can find in folder "videos" among the material posted this week.

4. Imagine you have the following code:

```
try {
   some_instruction_1
   some_instruction_2
   some_instruction_3

}
catch (Exception1 e1) {
}
catch (Exception2 e2) {
}
some_instruction_4
```

   and assume an exception is thrown while executing *some_instruction_2*. Is *some_instruction_3* ever executed? If the exception is not caught what happens? If the exception is caught in the catch block, is *some_instruction_4* executed? What happens if the exception if passed on to the caller?

5. Adapt the *Student* class for it to accept console input for the addition of a new *Student* object that does type checking and a minimum of input validation. You can use the *parseDouble()* or similar methods in the wrapper classes to input numeric values.

6. Create an array *Student[]* of N *Student* objects. The value N should be entered as input from the console. Ensure that the input is ok (you could use the static method *parseInt()*

of a wrapper class). Write then a method that asks for displaying a certain number M of objects from the array. Again ensure that M is ok and deal with the situation where M is out of bound (i.e. it is bigger or equal than N so that the index for accessing your array actually points to nothing) by defining your own exception class. You will need to throw and catch your exception objects when M is out of bound.

7. Write a program that allows entering two numbers a and b and displays the result of a/b. However, these must be the conditions:

    a- a and b are entered as numbers
    b- a and b must be integer values
    c- If either a or b are not integer your code must throw a *NumberFormatException* exception
    d- If b is zero your code must throw an *ArithmeticException*
    e- Anytime an exception is thrown, the message must be displayed also on the console