Lab 9 – the accruement

The mist-lit gray windows begin to seemingly catch on fire, turning golden-brown as the first rays of proper daylight pierce the haze. The diner lights still burn with wearied enthusiasm, clashing with the outer sunshine. Reflected in the window, dark circles hang beneath your eyes, a visual reminder of sleepless nights piling on top of each other relentlessly. The quiet jukebox adds to the general silence, a gloom penetrated intermittently only by the sound of fingers typing away – and the occasional loud sip from Bower.

The mischievous goblin of Insomnia had unfortunately passed her by, constant coffee binges being employed in warding off any potential drowsiness. Peters seems to be struggling harder, caffeine and all – her eyes closing for increasingly more seconds at a time behind the shimmer of her glasses. Locked in your private trance, you type away, keeping up the hypnotic rhythm of mechanical tap-tap-taps. Taking it up a notch wouldn't be the worst thing to do in your position. When all is said and done, you could all use a few hours off. Even yourself? Not before you've worked out all the minute touches in your application. Not that the rest would be of any use at this point. So you type on. And hold your breath in expectation. It's all any of you three can do for now, as you wait for the storm to brew and for sleep to truly become a luxury.

Continue your previous assignment, add the following features:

* Use exceptions to signal errors:
  + from the repository
  + validation errors – validate your entities using Validator classes
  + create your own exception classes
* Store mylist in a CSV(Comma Separated Values) or HTML(Hypertext Markup Language) file, depending on what you get as input for mylistLocation.
  + the CSV file will contain each entity on one line and the attributes will be separated by comma (,)
  + in the HTML file you will have a table, in which each row holds the data of one entity. The columns of the table will contain the names of the data attributes. You may find an example of a valid html structure here: https://www.cs.ubbcluj.ro/~iuliana/oop/Laboratories/Lab8-9.pdf
* Open mylist with the correct application:
  + CSV file – with Notepad, Notepad++, Microsoft Excel or OpenOffice Calc
  + HTML file – with a browser
* Create a UML diagram for your entire application. For this, you can use any tool that you like (StarUML is an example of open source software for UML). Do not draw the diagram by hand.

All commands must be in the form:

* mylistLocation fullPath (e.g. mylistLocation c:\some really long\path\with spaces\myFileName.XXX) - use the file extension given (XXX) to save in the appropriate format (.txt or .csv or .html)

Example test run: (make sure to create the files at the provided path)

1. call: fileLocation c:\some really long\path\with spaces\myFileName.txt
2. call: mylistLocation c:\some really long\path\with spaces\myfile.csv
3. call: mode A
4. call: add [valid input]
5. call: mode B
6. call: save [id from valid input]
7. call: exit

Check for [valid input in specified format, in c:\some really long\path\with spaces\myfile.csv]

**Bonus (0.2 p)**

In addition to the file-based implementation for the repository, implement an SQL-backed repository. For this, useinheritance and polymorphism. Choose any type of database management system (e.g. MySQL, SQLite, PostgreSQL).To receive the bonus, the requirements must be implemented correctly, by week 11 and the application must function properly.