Lab Work 2

Modeling Publications in Python

You are asked to model *Publication*s, specifically *Periodical*s and *Book*s in Python. As you might imagine, *Periodical*s and *Book*s, as types of *Publication*s have much in common -- but also have certain unique properties. You should capture this aspect of their nature using subtypes and inheritance.

All *Publications* have *title*s, a specific *number Of Pages* and a *publication Date*. They also keep track of the *number Of Times Read*. Additionally, *Book*s have authors and *ISBN*s, whereas *Periodical*s have *Volume*s and *issue Number*s.

Given the classes of objects, as described above:

- Initialize any type of publication, providing the properties appropriate to the type of publication, using a standard constructor
- Get a string representation of any type of publication, using the standard method for such
- Determine how many of any type of publication are in existence, as well as how many of all types of publications are in existence. This should work, even after some are deleted. It should use an appropriately named "get method", e.g. "getBookCount()"
- Get any property of any type of publication using a method named using the convention "getProperty()", e.g. getAuthor()
- It should be possible to increment the number of times any publication has been read by any positive integer -- but not decrement it
- Methods must be implemented as static methods, class method, or instance methods as appropriate, and so designated, as appropriate.

Once you have implemented class specifications describing the classes as described above, you should instantiate and delete several different instances of various publications and exercise the methods you have written to demonstrate that your class specifications are correct. In the verification, consider to create a list of publication and to show an application of Polymorphism

Marking criteria

1.	Code can be executed without any error	2%
2.	Classes are defined correctly with inheritance	2%
3.	Testing logic has been implemented to verify the classes	2%
4.	Application of Polymorphism	2%
5.	Sufficient comments and indentations	1%
6.	Efficient logic	1%