Challenge: Books

Description: Write a Java application in NetBeans that utilizes a class written for the application called **Book**. The application creates instances of **Book** objects and utilizes its properties, methods, and fields.

Purpose: To gain experience in creating classes and writing a program that creates object instances of the class.

Requirements:
Project Name: <Pawprint>Books

For the project name, follow the same naming scheme used in the first challenge. The project name is to be comprised of your pawprint, with the first letter capitalized, followed by Books. For example, if the pawprint is abcxyz9 the project is to be named Abcxyz9Books.

Create a Java Enumeration called **Category** with the following values:

- ACTION, ART, BUSINESS, CHILDREN, DRAMA, FANTASY, HISTORY, HORROR, MYSTERY, ROMANCE, SATIRE, SCI_FI, SELF_HELP, OTHER, UNKNOWN

Create a class called **Book** that contains the following:

- Five private fields: A String called **name**, a String called **author**, a double called **price**, a Category called **category**, and an int called **version**.
- You should have created a Java Enumeration in the previous step, so the type Category should work for the variable **category.**
- A public static field called **numberOfBooks,** which is an int, and has a default value of 0.
- Three constructors that chain each other:
    1. The first constructor is the no-arg constructor which sets name and author to empty strings, and increments numberOfBooks by 1.
    2. The second constructor receives parameters name and author
    3. The third constructor receives name, author, price, and category.

- Constructor2 that receives just the name and the author sets the version to 0.
- Constructor3 that receives name, author, price, and category sets the version to 1.
- The constructors are to take the data they receive as parameters and use them to set the values of the corresponding fields in the class.
- You must use the keyword “this” to differentiate between the class fields and parameters with the same name.
- Instead of setting the same values for constructors 2 and 3 (having the same lines of code), you should chain all three constructors. Therefore, constructor3 will take four parameters and chain with constructor2. The first line in constructor3 will call constructor2 passing name and author, then it will set price, and category class fields with the parameters sent. Constructor2 will take two parameters and chain with constructor1. The first line in constructor2 will call constructor1, then it will set name and author class fields with the parameters sent. The class field version (version variable) will be set with a default value in

the constructor not a parameter sent to the constructor. The default values are stated above.

- For help with chaining, you can look at the class notes, which includes a similar example.
- The following methods:
    - **setName**
        - Receives a book name and sets the name field using it.
        - Increments the version field value by 1.
        - Does not return anything.
    - **setAuthor**
        - Receives an author and sets the author field using it.
        - Increments the version field value by 1.
        - Does not return anything.
    - **setPrice**
        - Receives the price and sets the price field using it.
        - DOES NOT increment the version field value.
        - Does not return anything.
    - **setCategory**
        - Receives the category and sets the category field using it.
        - Increments the version field value by 1.
        - Does not return anything.
    - **getName** - Returns the name field value.
    - **getAuthor** - Returns the author field value.
    - **getPrice** - Returns the price field value.
    - **getCategory** - Returns the category field value.
    - **getVersion** - Returns the version field value.
- **NOTE**: You must use the keyword "this" in order to distinguish between the parameters sent to these methods and the class fields in the Book class. By doing this, you can use the same name for the parameters as the class fields.

In the main() method of the main class do the following:

- Create an instance of Book, assign the instance to a variable named book1, and supply a name of "Grant" and an author of "Ron Chernow" during creation in the constructor's parameters.
- Use the book1 instance and the setPrice() method to set the price to 22.00, and the setCategory() method to set the category to HISTORY.
- Create an instance of Book, assign the instance to a variable called book2, and supply a name of "Goodnight Moon", an author of "Margaret", a price of 8.94, and a category of CHILDREN during the creation in the constructor's parameters.
- Last, create an instance of Book, assign the instance to a variable called book3, and do not supply any parameters to the constructor by calling the no-arg constructor.
    - Use the book3 instance and the setName() method to set the name to "The Martian"
    - Use the book3 instance and the setAuthor() method to set the author to "Andy Weir"
    - Use the book3 instance and the setPrice() method to set the price to 15.63

  - Use the book3 instance and the setCategory() method to set the category to SCI_FI
- For book1, book2, and book3 print the name, author, price, category, and version to standard out. Use the get methods to get the name, author, price, category, and version to print.
- Precede the info that is printed with a line that says “Book 1:” before the book1 information, “Book 2:” before the book2 information, and so on.
- Place an empty line between the book1, book2, and book3 information so there is at least one space in-between each of them.
- Place “Name: “ before the name, “Author: “ before the author, “Price: “ before the price, “Category: “ before the category, and “Version: “ before the version.
- Place “Number of Books: “ before the numberOfBooks. You will print this at the end to show how many instances of Book were created. Since numberOfBooks is static, there is a particular way to call this field. Make sure you do it correctly.
- Place an empty line between the last book information and the number of books information, so there is at least one space in-between.
- The output should look like the following where the <> information is the field data obtained through the get methods.

Book 1:
Name: <name>
Author: <author>
Price: <price>
Category: <category>
Version: <version>

Book 2:
Name: <name>
Author: <author>
Price: <price>
Category: <category>
Version: <version>

Book 3:
Name: <name>
Author: <author>
Price: <price>
Category: <category>
Version: <version>

Number of Books: <numberOfBooks>

Challenge: Books

Run your application and make sure everything works as expected. ZIP the project directory using NetBeans (i.e. Export the project) and submit it on Canvas.

NOTE: If you would like to create additional books, then do so, these requirements are the minimum and creating more states, behaviors, and objects are usually rewarded. Have fun and experiment so you learn object-oriented programming and Java beyond the class requirements.

Things to submit on Canvas:

- The zip file created after you export your project from NetBeans.
- You may also submit screenshots of your application running for proof (optional). Put all your screenshots in a folder, name the folder "<Pawprint>Screenshots" where you replace <pawprint> with your pawprint, and zip them, even if you only take one. Then submit the zip of screenshots on canvas.

Note: You are only allowed to submit one thing at a time on canvas. You cannot submit a zip file and your screenshots. Therefore, first submit the screenshots, then click "re-submit", and submit your zip file. On your end, it will look like you only submitted the zip file, however, on our end, we will see both.