CET1022 - Data Structures and Algorithms with Java - Practicum 01

Topics Covered: Arrays

Learning Objectives:

- Familiarize with basic data structure - array, in particular the dynamic array where the size of the array is not fixed. This includes the implementation of:
    - lookup
    - insertion
    - deletion

Deliverables:

- Submit three Java files called - `Book.java`, `CustomBookArray.java`, `BookManagement.java` in a **ZIP** file using the format `CET1022_P01_<Trainee_name>.zip`. Eg: `CET1022_P01_John_Doe.zip`.
- Note that a non-working submission will result in a zero

Background

A new library have just been set up in a particular town area. Hence, they need a system to keep track of all the books they have in possession. The program can be used to add new books, delete books and also search for books. Apart from that, it shall also have features to display a list of all the books in their inventory or books starting with a particular alphabet.

Task

Your task is to implement from scratch a custom dynamic array called `CustomBookArray` class which will be used by the `BookManagement` class to store all the `Book`'s objects. The `CustomBookArray` will start with a **size of 2** to store the `Book`'s objects but the size shall be dynamic such that it can increase when more books are added to the array. However, the array size shall be limited to a **maximum of 8**.

Take note that the `Book` objects shall be stored in different arrays based on the starting alphabet of the book's title. So fundamentally, there should be 26 arrays, one for each alphabet, however, for brevity, let's assume that the library only have books **starting with the first 2 letters of the alphabet (case-sensitive)**. ie. aA or bB

IMPORTANT: You are **NOT** required and **NOT** allowed to use any third party packages or Java packages for this practicum other than the ones that are automatically loaded by the JVM (aka no `import` statements).

Refer below for details of implementation of each class.

`Book` Class

1. Your class should contain the following fields:
   - `title`
   - `author`
   - `page`
2. Initialize all 3 fields with external data within a single constructor.
3. Provide the appropriate accessor methods for the field members.

only the necessary accessor methods.

4. Each `Book` object must be printable with the following format:

```
Effective Java
    Author: Joshua Bloch
    Page: 375
```

it must be printable so you need to override the toString method.

`CustomBookArray` Class

1. Your class should contain the following fields:
   - `arraySize` - the size of the array
   - `arrayCount` - keep track of numbers of objects stored at any point of time
   - `limit` - maximum size of the array
   - `booksArray` - array containing `Book` objects
2. Initialize all fields with an appropriate value within a single constructor.
3. An `add` method which accepts a `Book` object.
4. A `delete` method which accepts in a `String` as argument.
5. A `searchBook` method which accepts in `String` argument.
6. A `getArraySize` method which return the array size.
7. Any other methods that you may require.

for the customBookArray class can have another methods, most likely you need a isEmpty and because it must be printable, you will need to override toString.

8. Each `CustomBookArray` object must be printable with the following format:

```
[Effective Java, Da Vinci Code, The Lost Symbol]
```

9. Requirements to take note:
   - Your array shall be able to resize once it reaches its capacity limit. Double the capacity every time the limit is reached when books are added.
   - Shrink your array capacity to the next minimum multiple of 2 after each deletion where appropriate.

you should be careful that if you shrink to 6 then you double to 12, make sure you limit it to 8 again.

`BookManagement` Class

1. Two `CustomBookArray` arrays initialized via constructor.
2. An `add` method accepting a `Book` object.
3. A `delete` method with a `String` input.
4. A `display` method which prints the contents of both arrays with the following format:

```
A: [Apple, Airplane, Apollo]
B: [Banana, Burger]
```

this display method does not require any inputs but the depends method in point 5 needs an input of the chosen book array. since you need 2 display methods, use method overloading.

5. Another `display` method that prints the contents of the chosen array of books using the same format in point 4.
6. A `searchBook` method which searches the arrays based on the given book title.
7. A `getArraySize` method which returns the size of the chosen array.

Sample Output:

Below is some sample output for your reference only.

```
// Sample input #1
// Input:
// 1. Title: "Apple Invasion", Author: "Tim Hill", Pages: 200
// 2. Title: "Bambi", Author: "Felix Salten", Pages: 150

//Sample output #1
A: [Apple Invasion]
B: [Bambi]
```

```
// Sample input #2
// Input:
// 1. Title: "animal farm", Author: "John Doe", Pages: 100
// 2. Title: "batman returns", Author: "Alfred Hugh", Pages: 350

//Sample output #2
A: [Apple Invasion, animal farm]
B: [Bambi, batman returns]
```

Rubrics

For the maximum allocation of marks, refer to the table below.

Description	**Marks (%)**
Successful implementation of the `Book` Class.	10
Successful implementation of the `CustomArrayBook` Class.	35
Successful implementation of the `BookManagement` Class.	10
Program is well tested (passes instructor's test case, no bugs, does not crash).	30
Proper and sufficient comments to explain code using Javadoc style	10
Proper display outputs (easy to read, ~~correct decimal places~~, etc.)	5