CSCI 2540 Assignment 4

100 points

Due date: Thursday, Sept. 29, by 11:59pm

In this assignment, you will write a program to simulate an inquiry system of a small library. You will need to write four classes: **Book**, **BookSearch**, **BookIdAlreadyExistException**, and **BookNotFoundException**.

The program should operate as follows:

First, read the library catalog from an input data file and store them into an array of Book type. The data file should be named **assg4_catalog.txt**.

In the input file, there is one line per book, and these lines have the following format:

```
        bookId          title           isbn            author          category
```

The *bookId*, *title*, *isbn* and *author* are strings while *category* is a character ('F' if the book is fiction; 'N' if it is a non-fiction book). Each column is separate by a TAB key. For simplicity, assume each column is only one word. If the book title includes multiple words, use "_" to connect them. A sample file is posted on Canvas.

You need to create an array of **Book** type to store all the books. While reading each book, if a bookId already exists, the program should throw an *BookIdAlreadyExistException*. Your program should handle this exception by printing a message and then skipping the rest of the line and continue reading from the file.

Once the program finishes reading, it should display all the books (except the ones with book id already existing) and print the total number of books in the catalog.

Next, read from standard input a customer's inquiry with a given bookId. If the book is found, it prints the information of the book. The output should include the bookId, title, isbn, author, and category ("Fiction" or "Non-Fiction"), printed on a single line. If the book is not found, it will print an error message. In either case, your program should allow the customer to continue to inquiry about other books. When the user enters "STOP" for bookId, it means the end of the customer's inquiry.

You need to write two exception classes: **BookIdAlreadyExistException** and **BookNotFoundException**. Both should be defined as checked exceptions. Each class should include two constructors. One is the default constructor, and the other is a one-parameter constructor and the parameter type is String.

Program Structure:

Your source code should be developed in four files: **Book.java, BookSearch.java, BookIdAlreadyExistException.java, and BookNotFoundException.java.**

Book.java will contain the class definition for a book according to the requirements specified below. ***BookSearch.java*** will be the application program with *main* method that reads from input file, stores the data into an array, and runs the simulation of the inquiry. It should also include exception handling code. ***BookIdAlreadyExistException.java*** and ***BookNotFoundException.java*** will define the two types of exceptions.

Each catalog item should be an object of the *Book* class. Define a separate instance variable of the appropriate type for the five pieces of information about each book. Instance variables should be maintained as *private* data. **Only one constructor with five parameter is needed**. Besides the constructor, the following methods are required for the Book class.

- The get method for each instance variable.
- The *toString* method that takes no parameter and returns all the information of the book as a combined string, including bookId, title, isbn, author, and “Fiction” or “Non-Fiction” for category.
- A static method *bookSearch* that takes three input parameters: (1) an array of *Book* objects that represent the entire catalog; (2) an integer specifying how many books are actually in the array; and (3) a string representing a bookId. The method should search the array of books looking for the book with the given bookId as specified by the third parameter. The method should return the index within the array. If it cannot find the item, the method should throw a *BookNotFoundException* but it is not handled in this method. Instead it will be handled in the main method where it is called.

Sample Catalog file:

```
A10001      Emma  0486406482    Austen F
L12345      My_Life      0451526554    Johnson      N
D21444      Life_Is_Beautiful     1234567890    Marin  F
A10001      West_Story    0486406483    Baker  F
C11111      Horse_Whisperer     1111111111    Evans  F
R25346      Japanese_Dictory     1234123488    Moon  N
```

Note: if you use Eclipse, the input file should be placed in your Java project folder (i.e., outside of *src* folder).

Technical Notes

- At the top of EVERY Java file you create, you must include a comment which states your name and what the program does. Follow the standard "Javadoc" commenting style (/** ……. */) for formatting your comments.

- YOU MUST PROVIDE COMMENTS for EVERY method that is included in your class. Follow the standard "Javadoc" commenting style (/** ……. */) for formatting your comments and include @param for each parameter and @return if there is a return type that is not void. You should also include Javadoc comments before each constructor.

- You do not need to finish writing the entire class before you can start testing it.
- Programs that do not compile will receive an automatic grade of "F".

Submission instructions:

You need to submit your programs electronically on Canvas.

Please **use a named package for each of your assignment.** For example, for assignment 4, create a new package and **name your package as assg4_yourPirateId** (use lower case for your pirate id), such as assg4_smithj21. You also need to include a statement such as “package assg4_smithj21;” at the beginning of each of your .java file (it will be automatically generated in Eclipse if you create the class inside the given package). **Please follow this naming convention exactly for all future assignments. You will be deducted points for not doing so.**

When you submit your files to Canvas, please submit a zip file with your package folder inside the zip file. The package folder should include only .java files (make sure you include .java files, not .class files). The name of the folder should match with your package name. (You can use 7-zip software to zip files/folder). **Once your zip file is unzipped, it will generate a folder that matches your package name.**