CSD 3464 –Test 01 (Question 01)

Overview:

The following question evaluates your knowledge of *Absolute Java* (6th Ed.)'s Chapter 1- 5 material. Please follow the instructions included in **this** document and implement the following Java files:

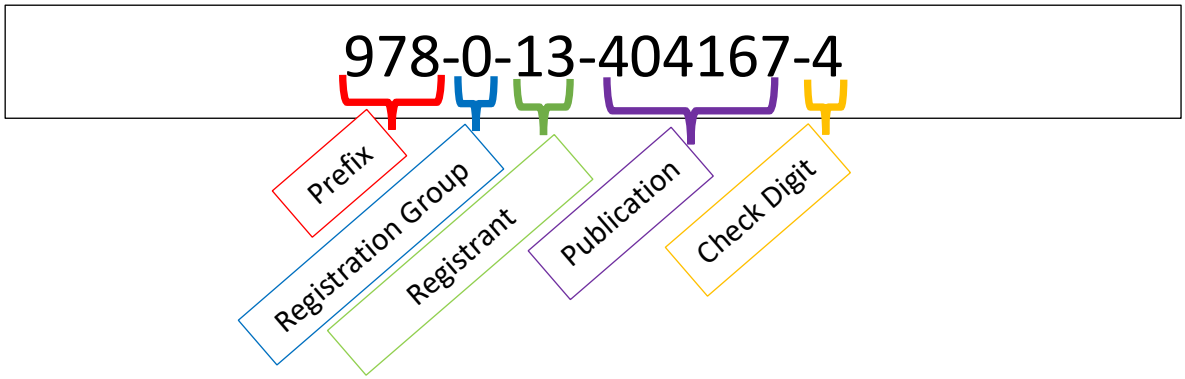
- ⇒ Course.java
- ⇒ Textbook.java
- ⇒ Isbn.java

The above classes/files should be inside a package called *q1*.

Instructions:

Step 01 – Isbn.java

Your first task is to implement an `Isbn` class that can represent 13 digit ISBNs; for example, the 13-digit ISBN for the course textbook, *Absolute Java* (6th Ed.), is shown below. A brief description of each ISBN component is also included in the following table for your convivence.



ISBN Component	Number of Digits	Description
Prefix	3 digits long	The prefix will change over time, but currently only the prefixes of "978" or "979" have been used.
Registration Group	1-5 digits long	Represents country and language information of the publication; for example, books published in the United States use "0", while other English speaking countries use "1". Countries with lower publication volumes are given longer Registration group codes; for instance, Nepal has the 5 digit code "99946".
Registrant	2-7 digits	Publishers are given a unique registrant code. Those publishers who publish more material have smaller registrant codes; for instance, our course textbook's

		publisher is Pearson which publishes a lot of textbooks and materials and therefore has the 2 digit code "13".
Publication	1-6 digits long	Each title and edition of the book will have its own distinctive number; for instance, the publication code for *Absolute Java* (6th Ed.) is "404167", while the previous edition has the code "283031".
Check Digit	1 digit	This ISBN component is used to check all the other components are valid. It does this through the use of a modulus algorithm that will be discussed later.

The class must have fields that hold a `String prefix, String registrationGroup, String registrant, String publication, String checkDigit, String seperator.`

- **Note:** The `String seperator` will either be the value `" "` or `"-"`. This is `String` is what is placed between each component of the ISBN when printing.

The `Isbn` class should have three constructors as listed below. In both constructors you must ensure the restrictions in the table above correctly enforced; for instance, the `prefix` mut be 3 **numeric** characters long and be either the values `"978"` or `"979"`. Also, don't forget to ensure only `" "` or `"-"` are used in the ISBN as a `separator`, but not both, and the numeric characters in the `String` are 13 total.

1. The first constructor should be a one-parameter constructor that takes a `String isbn`. The `String isbn` should consist of the 5 components listed above separated by either spaces or hyphens; for instance, "978 0 13 480221 3" and "978-0-13-480221-3" are valid ISBNs for the textbook titled *Starting Out With Java* (7th Ed.).

2. The second constructor should take 6 `String` values: `prefix, registrationGroup, registrant, publication, checkDigit,` and `seperator.` Just as in the above constructor, constructor #1, you should ensure each ISBN components are assigned only valid values.

3. The third constructor required by the `Isbn` class is a **copy constructor**.

(Very) Helpful Hints:

- Implement a `static boolean isValidISBN` function that takes an ISBN as a `String`
  - Return `true` if valid or `false` otherwise
- Start by checking the `String length()` is:
  - 13 digits + 4 `seperators` = **17 TOTAL characters**
- If the `length()` is 17, confirm the `String` contains either 4 " " or "-", but not both.
  - You may want to for loop over your string and count the occurrences of " " and "-".

- After confirming the appropriate `separator` you can use `StringTokenizer` to split the `String` into 5 tokens.
  - Loop over **EACH** of the first 4 of 5 tokens and ensure that each token is the appropriate length and/or values from the table provided above. If a valid value is provided you should update the appropriate instance variable.
    - **IMPORTANT:** To check the each token only contains digits you should loop over each character in the token and call `Character.isDigit()` to return a `boolean` result indicating if a character is a digit or not.
  - The 5th token should only be a single digit. We also need to verify its correctness by calling the `private int` method `calculateCheckDigit`.
    - If the result of `calculateCheckDigit` matches our 5th token we may assign its value to the instance variable `checkDigit`.

In terms of methods, the `Isbn` class should have a `public static` method `calculateCheckDigit` which takes a `String isbn` as a parameter. This method is responsible for performing the below calculation:

Let

$$r = \left(10 - \left(x_1 + 3x_2 + x_3 + 3x_4 + \cdots + x_{11} + 3x_{12}\right) \bmod 10\right).$$

Then

$$x_{13} = \begin{cases} r & ; r < 10 \\ 0 & ; r = 10. \end{cases}$$

For example, given the ISBN 978-0-13-404167-4:

r = (10 – (9 + 3 x 7 + 8 + 3 x 0 + 1 + 3 x 3 + 4 + 0 x 3 + 4 + 1 x 3 + 6 + 7 x 3) mod 10)
= (10 – (86) mod 10)
= (10 – 6)
= 4

Since r < 10, the checkDigit (x_{13}) is therefore 4

NOTE: That we have correctly calculated the as the 13th digit in the ISBN 978-0-13-404167-4 is indeed 4

Your class must also include the method `public equals()` which compares two `Isbn` objects and returns the `boolean` value `true` if and only if all the `Isbn`'s instance variables, excluding the `separator`, are the same.

- This means our function would treat ISBNs "978-0-13-404167-4" and "978 0 13 404167 4" would be considered the same.

Additionally the class must include the method `public toString()` which returns the five components of the ISBN separated by the `seperator`.

Lastly, please ensure the class includes an accessor for each instance variable, as well as a `public setIsbn` which takes a single `String isbn` and updates the "this" `Isbn` object only if a valid `isbn` is provided.

- You may use the `static boolean isValidIsbn` to confirm the validity of the `isbn` before updating the "this" object

Step 02 – Textbook.java

Create a `Textbook` class with `private` fields that hold a `String title, String publisher, Isbn isbn, int edition,` and `int numPages`.

The `Textbook` class should have three constructors:

1. The first should be a no-parameter constructor which sets the fields `title` and `publisher` to `"TBD"`. `isbn` should be set to a `null` value. Lastly, you may set `edition` and `numPages` to 0.

2. The second constructor should take three parameters `String title, String publisher, String isbn, int numPages,` and `int edition.`

    a. `title` and `publisher` should not be empty strings (if illegal values are provided assign the default value `"TBD"`)
    b. `isbn` should be valid (if illegal set value to `null`)
        - Recall: `Isbn` has a static Boolean `isValidIsbn` method
    c. `numPages` and `edition` should be an integer value of 1 or greater (if illegal values are provided assign the default value `1`).

3. The third constructor required by the `Textbook` class is a **copy constructor**.

Your class must also include the method `public equals()` which compares two `Textbook` objects and returns the `boolean` value `true` if and only if all the `Textbook`'s `isbn` are the same.

- If `isbn` values of both `Textbook`s are `null`, return `true` if the `Textbook`s have the same `title` and `edition`.

Additionally the class must include the method `public toString()` which returns a String in the format:

TITLE: `title`
PUBLISHER: `publisher`
ISBN-13: `isbn`
EDITION: `edition`
PAGES: `numPages` pgs.

- If `isbn` is `null`, display "`TBD`" in the returned String value

Lastly, please ensure the class includes an **accessor and mutators** for each instance variable. Be sure that instance variables are only updated if valid values are provided, leave them unchanged otherwise.

Step 03 – Course.java

Create a `Course` class with `private` fields that hold a `String name, String courseCode, String description, Textbook txtBook` and `double credits.`

The `Course` class should have three constructors:

1. The first should be a no-parameter constructor which sets the fields `name` and `courseCode` to "`TBD`". `description` should be set to " ". `txtBook` should be `null`. Lastly, you may set `credits` to 0.0

2. The second constructor should take five parameters `String name, String courseCode, String description, Textbook txtBook` and `double credits.`
    a. `name` and `courseCode` should not be empty strings (if illegal values are provided assign the default value "`TBD`")
    b. `credits` should be an floating value of 0 or greater (if illegal value is provided assign the default value `0`).
3. The third constructor required by the `Course` class is a **copy constructor**.

Your class must also include the method `public equals()` which compares two `Course` objects and returns the `boolean` value `true` if and only if all the `Courses`' `courseCodes` are the same.

Additionally the class must include the method `public toString()` which returns a String in the format:

```
COURSE: name
CODE: courseCode
CREDITS: credits
DESCRIPTION: description
===TEXTBOOK===
txtbook.toString()
```

Lastly, please ensure the class includes an **accessor and mutators** for each instance variable. Be sure that instance variables are only updated if valid values are provided, leave them unchanged otherwise.