OBJECT-ORIENTED PROGRAMMING
INDIVIDUAL PROJECT 01

Project 01 Background

This is an individual programming assignment. It is not to be completed as part of a team or peer programming partner.

Your goal is to create a program that is made up of four classes:

- **`CSC211Project01`**
- **`StudentDriver`**
- **`Student`**
- **`Textbook`**

The Textbook Class

The **`Textbook`** class represents a university-level textbook. Textbooks have the following characteristics:

- **`subject (String)`** - subject to topic area covered by the textbook. The default subject area is Object-Oriented Programming.
- **`pageCount (int)`** - number of pages in the textbook. Default number of pages is 800.
- **`unreadPages (int)`** - number of pages that have never been read. The default number of unread pages when a textbook is created is the same as the number of pages. (You just got the book, you couldn't have read any of it yet.)

A **`Textbook`** can be created (instantiated) three different ways:

- No Formal Parameters - In this case, the default values should be used for the data fields.
- With a subject only.
- With a subject and number of pages.

A **`Textbook`** has the following accessor methods:

- **`String getSubject()`**
- **`int getPageCount()`**
- **`int getUnreadPageCount()`**
- **`double getWeight()`** - Weight (in kilograms) is determined by multiplying the number pages in the textbook by the weight of each page (**`0.0025 kg`**).

The **`subject`** and **`numberOfPages`** for a **`Textbook`** object cannot be changed (mutated) — unless of course you rip pages out of the textbook, but we're going to treat our books with respect and not do that.

The number of unread pages changes when we read pages. Reading pages of a `Textbook` object creates knowledge. to read pages and create knowledge, a `Textbook` has the `readPages` method:

- `int readPages(int numPages)`

  The formal parameter is the number of pages the client wants to read. The `readPages` method has two requirements:

  - If `unreadPages` is ***not*** zero, then the knowledge created is `PAGE_KNOWLEDGE` times the `numPages` read. `unreadPages` should be decremented by `numPages`, but should not go below zero. (`PAGE_KNOWLEDGE` is 5 per page for university-level textbooks).
  - If the number of unread pages is zero (the entire book has been read) then the knowledge created per page is halved because the pages are being reviewed.

  Optional goals for the `readPages` method:

  - Validate that `numPages` is greater than or equal to zero.
  - If `numPages` is greater than `unreadPages`, knowledge should be:

```
unreadPages * PAGE_KNOWLEDGE + (numPages - unreadPages) * PAGE_KNOWLEDGE / 2.0
```

The Student Class

The `Student` class represents a university student and should have the following characteristics (data fields):

- `name (String)` — name of the student. Default is “`Pat Zhang-Garcia`”
- `book (Textbook)` — the `Textbook` object the student is carrying.
  Default is “`Object-Oriented Programming`” with `800` pages
- `health (double)` — the student’s health. Initial value is ***always*** `1.0`;
  Range is `0.0 <= health <= 1.0`
- `knowledge (int)` — student’s knowledge. Initial value is ***always*** `0`;
  Minimum: `0`; Maximum: there is none!

You only need to have a default constructor, though you can create more if you like. `health` and `knowledge` should not be a parameters in a constructor since they are always initialized to the same values.

All data fields should have accessor (getter) methods. Only `name` should have a mutator (setter) method.

Student’s have a learnSomething method with the following signature:

```
public void learnSomething(int numPages)
```

When the `learnSomething` method is called, it should read `numPages` pages in `book` by calling the book’s `readPages` method and increase the student’s `knowledge` by the amount returned from the `readPages` method.

The StudentDrive Class

This class should thoroughly test your Student class. It must have at least one method, **`testStudent.`** You can decide on the data fields you need, the constructors and other methods.

You can implement the **`testStudent`** method as a canned (pre-programmed) test of the **`Student`** class or as an interactive test.

The CSC211Project01 Class

This class should instantiate an **`StudentDriver`** object and call the **`testStudent`** method.