This assignment is due by **10pm on Monday September 18** and is worth 16 points.

1 Goals

The goal of this assignment is to get some practice using and designing classes, and more experience working with Java.

2 Your Assignment

This a solo assignment - you should not work with a partner. The reason for this is to make sure that everyone feels comfortable working with Classes in Java on their own. All of your code files should include your name at the top (in comments). You can ask questions on Piazza or talk to me, the lab assistants, or the prefect if you're having trouble. The appendix of your book is also very helpful for Java syntax. Moodle has a link to the Javadocs for Java - these are incredibly useful as a reference! (I often refer back to Javadocs when writing code - that's what they're there for!)

For this assignment you'll be writing two programs (or classes): a Book class that stores information about a particular book and a Library class that uses the Book class. You'll get some practice thinking about static vs. non static variables and methods, creating and accessing instance variables and a little bit of practice with using arrays in Java.

2.1 Book Class

Create a Book class (inside a file `Book.java`) that stores information about a particular book. In particular, your class should have the following features.

1. Your class should have the following instance variables:

   `String title` the title of the book
   `String author` - the author of the book
   `int pubYear` - the year in which the book was originally published.

2. Your class should also have the following ***static*** instance variable:

   `int CUR_YEAR = 2017` - this represents the current year.
   Why would you make this a static variable? Answer this question in comments in your code.

3. You should have a constructor with the following signature:
   `Book(String aTitle, String anAuthor, int aYear)` that specifies the title, author and publication year. You may create other constructors as well, but you must have this constructor.

4. Add appropriate accessor (getter) methods.

5. Also, provide the following two methods:

- `getAge()` - This method should return the age of the book in years. That is the number of years between when it was published and the year stored in the variable `CUR_YEAR`.
- `toString()` - This method should return a string with information about the book. For instance, if you had a book object for the book Emma by Jane Austen, published in the year 1815, this method should return the following String:

  "Emma by Jane Austen is 202 years old."

6. Add a `main` method to your class that creates two different Book objects and prints out some information about them when the program is run from the command line. For example, compiling and running my program produces the following output:

```
$ javac Book.java
$ java Book
Emma by Jane Austen is 202 years old.
Pride and Prejudice by Jane Austen is 204 years old.
```

7. Add another instance variable of your choosing to your class and create a second constructor that takes in 4 parameters, including information for your new instance variable. Be careful, your original constructor should still work when only given 3 parameters.

2.2 Library Class

You will now create a Library class that utilizes your Book class. There are lots of ways that you could create and use this class. For this assignment, you will create two methods in this class:

- Create the following **static** method: `public static Book oldestBook(Book[] lib)`. This method takes one parameter, an array of Book objects `lib` and returns a reference to the Book object in the array that is the oldest (the most number of years have passed since publication). Why would you make this a static method? Answer this question in the comments for this method.
- Create a `main` method that creates an array of at least 3 different Book objects and print information about all the books to the screen (I suggest you use the `toString()` method from the Book class here). Then use the `oldestBook` method to find the oldest book and report which book it is. For example, compiling and running my program produces the following output:

```
$ javac Library.java
$ java Library
My libray contains the following books:
Emma by Jane Austen is 202 years old.
Pride and Prejudice by Jane Austen is 204 years old.
The Old Man and the Sea by Ernest Hemingway is 65 years old.

The oldest book in the Library is: Pride and Prejudice
```

- **Extra Challenge (not required)** - Update your Library class to also compute some additional information about your library (e.g. the number of different authors, list books alphabetically, etc.).

2.3 Final Hints

- Look back at examples from class or your book if you get stuck. It's totally okay to experiment and try things out to see what will happen - that is all part of the learning process.
- Make sure to save your work regularly as you go along!
- If you get stuck, don't hesitate to ask for help from a lab assistant, prefect or me!

3 Submission and Grading

You'll submit `Book.java` and `Library.java` to Moodle as a zipped file. Specifically, put these files into a directory named `[your_last_name]HW3`, zip this directory, upload it to Moodle. For example, my directory would be named `OesperHW3` and the resulting file would be `OesperHW3.zip`. See HW2 for more information on how to zip up your files.

3.1 Grading

This assignment is worth 16 points. Below is a partial list of the things that we'll look for when evaluating your work.

- Does the program compile? It is essential that you check whether your submitted work compiles. If you submit a program that doesn't compile, you will automatically receive at least a 25% deduction for the assignment, even if the problem was relatively minor.
- Running Book from the command line prints out the specified information for two different book objects.
- Running Library from the command line works as specified above.
- A constructor, `getAge` and `toString` are all defined in the Book class.
- Comments are used to indicate the logic of the program.
- Program has your name and a short description of the class at the top of each class in comments.
- Program uses correct indentation.
- Program follows typical Java style rules as outlined in the Style Guidelines on Moodle.
- Program uses an appropriate object oriented design, including good choices about what is static and what is non-static, what is private, etc.

Start early, ask lots of questions, and have fun! Layla, the lab assistants, and the prefect are all here to help you succeed - don't hesitate to ask for help if you're struggling!