Programming Project - Sorting

For **all** parts of the assignment below you should write a method that either uses or modifies a sorting method from the example sorting class (Sorting.java) we did in class. Do not use any Java library sorting methods. For parts 3 & 4 your solution should use sorting in order to receive credit.

1) For this project, you should first extract the code in Project7StarterCode.zip.
2) Add the following methods to the Sorting class in the csc205 package:
   A) `sort(array)` should take an array of Comparable objects as a parameter and sort it in place.
   B) `cutoff_qsort(array)` that implements a quick sort, but once the partition reaches a small enough size, uses another sorting technique that is more efficient with small sizes. `cutoff_qsoft(array, n)` that uses n as the cutoff size
   C) `find_nth(array, n)` that **efficiently** finds the element in the array that would be in the nth position if the array was sorted. You should not assume the array is sorted and your method should be quicker than simply sorting the array and returning the nth element. (Note: there are multiple ways to implement this. The most efficient way is to modify one of the sort methods we discussed.)
   D) Create a method called `closest_pair()` which accepts an integer array as a parameter and returns the two integers in the array that are closest together. If there is a tie, return the smallest pair.
3) Create a method called `shuffle_array()` in the Deck class which shuffles an array using the method we discussed in in class (sorting an array of random numbers). Do not use the built in Java shuffle methods or any other shuffling algorithm.
4) Your methods and code should work with the provided sample driver **with no modifications**. If your code does not work with the sample driver you may receive a grade of 0 for that part. Your code will be tested using a test driver similar to the sample driver.
5) You may reuse the existing sorting methods where appropriate. Unnecessarily duplicating existing code will result in a grade of 0 for that method. Feel free to implement helper functions or additional classes where appropriate.
6) Create an implementation document that includes **sample output** from the sample driver and the sorting methods you implemented along with a brief discussion of your implementation. Include in your discussion any alternate implementations you tried and give the reasons you decided not to include them. Also include your shuffle_array() method and sample output showing a sorted array of cards.

Submission requirements:

- **Include your name** as a comment at the top of each source code file
- Make good use of whitespace/comments to make your implementation clear.
- Your writeup should be a well-formatted .doc, .pdf, or .txt file. **Points will be deducted for documents that are not clearly formatted.** This includes not using a fixed-width font for code, unnecessary vertical whitespace, and any formatting that reduces readability.
- Zip your entire Eclipse project (including .class files). Do not use .rar.
- **Include your first and last name in the .zip filename**
- Upload your implementation/output document & zipped project **separately** to Canvas.
- (Optional) Submit a hard copy of your implementation document.

Your submission for this project should be your own work. You should make every effort to figure out how to write these methods on your own. If you get stuck, ask me - **do not rely on the internet or another student**. The final exam will feature questions based on this project so it is essential that you are able to figure out the solution **on your** own and that you fully understand what you submit. If you get stuck, ask!

The implementation document will make up 30% of the grade for this project. Submit a hard copy or submit your document **separately** from your code on Canvas.