This introductory assignment is intended to be done by your team of two to three students. You may collaborate on answers to all questions or divide the work for the team. In any case, the team should review the submission as a team before it is turned in.

Part 1: OO Concepts and Definitions – 10 points

Provide definitions for each of the following six terms. Remember to cite sources, even if the source is the textbook. You may want to look for OO books (including the textbook) on the CU Libraries link to O'Reilly-Safari e-books, under the Sciences tab here: https://libguides.colorado.edu/strategies/ebooks, please note that Wikipedia is not usable as a primary source, neither (in this case) are class slides.

- abstraction
- encapsulation
- polymorphism
- coupling
- cohesion
- identity

Along with the definition of each term, also include:

- an explanation of how each term applies to the object-oriented notion of a class, and
- provide an example of the term applied in the design of a class or a set of classes (can be shown as code, psuedo-code, text, or graphic examples).

Part 2: Java exercises – 15 points

These coding exercises are designed to let you familiarize yourself with basic Java and ensure your GitHub and Java development environments are set up appropriately. You should use a Java 8 or later environment to develop this code.

1. Create a Java program that loops through the following functions:

- reads a string from the console input
- sorts the letters in the string into alphabetic order
- prints the sorted string to the console output

Use an instance of a class called Sorter with three methods: read, sort, and write to perform each operation. An example of input and corresponding output would be "Bruce" and "BCERU". The program will end if a null string is submitted as input. Run the program using the full name of each team member as input and capture the output to a Results1.txt text file (output capture can be done in any fashion, including just by cutting and pasting output from the console into the file).

2. Create a Java program that begins a loop that:

- asks for a numeric positive integer range (a start and end value)
- reject any range where the end value is less than or equal to the start, or if negative numbers or floating point numbers are provided as input
- if both values provided are 0, the program should end

- create an array of ten random integers in the range entered (including the start and end values provided as possible results)
- print the array of numbers followed by the mean and standard deviation of the array values
- extend the size of the array by adding 10 more random values and recalculate and print the values and the mean and standard deviation
- this should continue until the program has reached an array size of 100; at that point after the printed result, the program should ask for the next start and end value

Capture the output to a Results2.txt text file. Run the program for the following start and end values:

- 1,2; 2,2; 4,2; -2,2; 0.5,0.8; 1,10; 1,100; 0,0

A typical output might look like (for the first 2 passes at the 1,2 range):

Values: 1 2 2 1 1 1 2 1 2 1
Mean 1.4 SD 0.516397779

Values: 1 2 2 1 1 1 2 1 2 1 2 2 1 2 1 1 2 2 1 2
Mean 1.5 SD 0.512989176

Use an instance of a class called Ranger that has methods read, calculate, and write that will read the start and end values, perform the array creation, and write out the data with the mean and standard deviation. Use the standard Java random number and statistical functions for calculations.

3. Write a Java program that uses a logical expression that tests whether a given character code is a

- lower case
- upper case
- digit
- white space (like null, backspace, space, tabs, etc.)
- or a special character (like ! or >) in ASCII.

Document how you decide to segregate the ASCII codes into the categories. Your program should be able to read a list of ASCII codes and then output the results for each code. For each input ASCII code print out the input code, the type of character, and the ASCII character to the console (for capture to Results3.txt). A typical printed line would look like:

Code: 66 Type: Upper Case ASCII Char: B

Use an instance of a class called Decoder that has a method decode that will return the type and ASCII character representation for a given ASCII code.

Run it on the following ASCII codes: 66,114,117,99,101,32,83,97,121,115,32,72,105,33,7,9,50,48,49,57 and capture the output to a Results3.txt text file.

Grading Rubric:

Homework/Project 1 is worth 25 points total.

Part 1 is worth 10 points and is due on Wednesday 9/1 at 8 PM. The submission will be a single PDF per team. The PDF must contain the names of all team members.

Questions will be graded on the quality (not quantity) of the answer. Remember your answers require definitions, examples, and citations. A solid answer will get full points, missing elements or poor-quality answers will cost -1 point each, missing answers completely will be -2 points.

Part 2 is worth 15 points and is due Wednesday 9/8 at 8 PM. The submission will be a URL to a GitHub repository. The repository should contain the code for all three exercises, a captured text file with program results for each exercise (use the names Results1.txt, Results2.txt, and Results3.txt), and a README file that has the names of the team members, the Java version, and any other comments on the work. Only one URL submission is required per team.

Code should be commented appropriately, including citations (URLs) of any code taken from external sources. A penalty of -1 to -2 will be applied per exercise for poor or missing comments.

Each code exercise should include evidence of output from a run captured in the text file mentioned per exercise. A penalty of -1 to -2 will be applied per exercise for incomplete or missing output.

A README file with names of the team members, the Java version, and any other comments should be present in the GitHub repo. Incomplete/missing READMEs will be penalized -1 to -2 points.

Please ensure all class staff are added as project collaborators to allow access to your private GitHub repository. Do not use public repositories.

Overall Project Guidelines

Assignments will be accepted late for four days. There is no late penalty within 4 hours of the due date/time. In the next 48 hours, the penalty for a late submission is 5%. In the next 48 hours, the late penalty increases to 15% of the grade. After this point, assignments will not be accepted.

Use e-mail or Piazza to reach the class staff regarding homework/project questions, or if you have issues in completing the assignment for any reason.