This introductory assignment is intended to be done by you as a solo effort. The goal of the assignments is first, to review basic OO terms and second, to get set up to perform basic development tasks with Java.

Project 1.1: 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 web sources, open courseware, etc. OO books (including the textbook) can be found via search on the CU Libraries link, or in the 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 the class slides.

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

For each term above provide:

- the definition of each term (supported by citation)
- an example of the term applied in the design of a class or a set of classes – this can be shown as code, psuedo-code, text/description, or graphic/UML examples.

Capture your answers in a PDF with your name for submission to Project 1.1 on Canvas.

Project 1.2: Java exercise – 15 points

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

Program 1: Compare the results of using 3 random number generators using Java arrays.

- Enter a loop asking for an integer value, numRands, from the user in the console
  - When a user enters a null input (i.e. hits return with no number) the input loop (and the program) should end
- Generate 3 arrays of double values, each of length numRands, using the following random functions from these Java libraries:
  - Java.util.Random
  - Math.random
  - java.util.concurrent.ThreadLocalRandom
- At that point, apply these descriptive statistics methods against each array, including:
  - mean
  - standard deviation
  - minimum value
  - maximum value

    - Note that the calculations of these values **may not** use a math or statistics library (i.e. math.stddev(list)), you must write the code to perform each of the statistical operations on an array of numbers
- Print out to the console a table of these values for the data from each library's random function
    - Name of function, numRands, mean, std. dev., min, max
- Return to the loop asking for user input until ended
- There should be at least two classes: Creator, to create the random number arrays, and Analyzer, to perform the statistics analysis.

Capture the output console text from data entry and calculations (cut and paste to a text file is fine) from at least three test runs – say 10, 100, 1000.

Program 2: Create a Java program that implements the following puzzle creator using ArrayLists.

- Create a class called "Reader" with appropriate methods to:
    - Create an ArrayList of 6 words each of which must be at least 5 letters long and made up of only letters
        - These can be read from a constant array of the 6 strings
        - Ex: printer
    - Your code should convert each word into uppercase
        - Ex: PRINTER
    - Create an ArrayList of 6 clue strings corresponding to each of the 6 words above
        - These can be read from a constant array of the 6 clues in strings
        - Clue: "Device made to waste ink and paper"
- Create a class called "Tokenizer" with appropriate methods to:
    - Break each word into 2 letter tokens (if the word has an odd length, the last token will be three letters long) and place them on an ArrayList
        - Example: For input word "PRINTER"; output tokens should be 'PR','IN','TER'
    - Create a single ArrayList of all tokens created
    - Randomize the order of all the tokens from all words in the ArrayList
- Create a class called "PuzzlePrint" with appropriate methods to:
    - Print "Tokens", followed by the randomized tokens printed out as a table, 4 tokens per line, tab delimited (the last line printed may have less than 4 tokens in it)
    - Print "Clues", followed by a list of the entered clues
    - Print "Answer Key", followed by the original words entered in uppercase

Run the program based on a set of 6 words you select (including the word "printer"). Output from the code should be sent to the console and captured as a text file (cut and paste of output from the console is fine).

Coding/Submission Guidelines

Concatenate your Java source and captured output in a text editor or word processor and create the PDF to submit for Project 1.2. Be sure it is clear what each of the four sections are (Java Program 1, Results, Java Program 2, Results). Include your name in the submission.

All code should be appropriately commented and should include the author's name in the comments.

If you use code from any external source, say Stack Overflow for example, please cite your original source and include the URL in your comments.

Code in the Java main function for both programs should be limited to basic object instantiation and execution. All operations and information should be handled in the objects of the three classes above; for example:

```
public static void main(String args[])

{

	Reader read1 = new Reader();
	Tokenizer tkr1 = new Tokenizer();
	PuzzlePrint pp1 = new PuzzlePrint();

	read1.createLists();
	tkr1.createTokens();
	pp1.printPuzzle(tkr1.tokens, read1.words, read1.clues);

}
```

Grading Rubric:

Project 1.1 is worth 10 points total and is due on Wednesday 1/25 at 8 PM.

The submission for 1.1 will be your PDF with each of the six terms, their definitions, and your examples.

Questions for 1.1 will be graded on the quality (not quantity) of the answer. Remember your answers require definitions, examples, and citations (in any format). 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.

Project 1.2 is worth 15 points total and is due on Wednesday 2/1 at 8 PM

The submission for 1.2 will be a PDF containing the Java code for both programs and captured text output results for each program.

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 program should include evidence of output from a run as described per exercise. A penalty of -1 to -2 will be applied per exercise for incomplete or missing output.

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 office hours, 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.