**CSCI165 Computer Science II  
Lab Exercise  
Object Oriented Composition**

**This is part one of this lab. Expect more later this week**

**In UML Composition is represented by:**

**Lab Task One:** Define properly encapsulated Java classes to represent the following composition diagram. Use the Date class that was included in the code for this week.

A screenshot of a social media post

Description automatically generated**Privacy Leaks must be prevented**

**Domain Validation:**

* **Customer:**
  + **Email:** 
    - must contain an ***@*** and only one ***@***
    - a top-level domain of either ***2 or 3 characters***
    - The top-level domain must be at the end of the email
    - @ must occur **before** the domain
    - If email is invalid set the field to “none on file”
* **Address:**
  + **City, State:** These fields should be populated automatically by specifying a zip code. I have included an offline version of ***zip\_code\_database.csv*** or you could research and figure out how to talk to the USPS Web Tools API: <https://www.usps.com/business/web-tools-apis/welcome.htm>   
    Validate the zip code also . . . if it is invalid decide on a way to handle this. I can help you brainstorm. I am interested in your design choices.
* **Account:**
  + **Credit Limit:** Cannot be more than 200% of the balance. If it is, make it 200% of the balance.
  + **Balance, Credit Limit and Discount Level:** Cannot be negative
  + **Discount Level:** For every year the account has been active it gets a 2% discount.
* **Product:**
  + **SKU:** Must be 10-character length string starting with one of the following
    - 001, 002, 003, 004, 110
  + **Price:** Cannot be negative

**Unit Test:** Define unit tests for each of the requirements above. Be sure to include full coverage. If you are unsure of this, talk to me.

**Driver:**

* Define a Driver class that demonstrates that you can create instances of the Invoice class. Demonstrate that you can stack toString and equals calls. Ensure that there are no null pointers.
* Open **customers.txt** and build an array of 1000 Customer objects with this data. A postal code is provided, the city and state will need to be pulled from the zip code database.

**Lab Part 2: Application**

**Ask questions if anything is unclear**

**Products:** Create an array of 1000 products

* The products are listed in the file **products.txt** This is randomly generated data so expect the names and descriptions to not be logical. This is not relevant, and I do not want to debate it. These data are tab separated.

**Accounts:** Create an array of 1000 accounts

* **Customer:** Randomly assign from provided list. You can just go sequentially.
* **Date Created:** generate a random valid date within the last 15 years
* **Account ID:** Take the customer first and last name and
  + Remove all vowels
  + Convert the consonants to all upper case, concatenate together
  + Concatenate the date created with no slashes ***mmddyyyy*** pad with zeros if necessary
  + Calculate a ***check digit*** by summing the ASCII/Unicode values of the name consonants and modulus length\_of\_name\_with\_vowels\_removed
  + **Example:** Ken 03/06/2020 => KN030620201
    - K = 75, N = 78
    - 75 + 78 = 153
    - Length of “KN” = 2
    - 153 % 2 = 1 => check digit equals 1
    - Concatenate check digit to the ID
  + You do not have to worry about handling ID clashes
* **Discount Level:** Assign discount level based on date created
* **Balance:** Randomly generate
* **Credit Limit:** Initialize to 10% of balance

**Invoice:** Create an array of 1000 Invoice objects

* Implement compareTo on the amount due
* Randomly select accounts. They can be duplicated
* For each Invoice randomly add between 1 and 20 products
* Implement a selection sort on the Invoice array. Do not use an API sort method. Write your own
  + <https://en.wikipedia.org/wiki/Selection_sort>
  + <https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html>
* Once the array is sorted, iterate through the array showing the toString for each invoice. Pause the display at each invoice and allow for a button press to advance to the next display.
* **Submit:** push ***Invoice.java, Customer.java. Product.java, Account.java, Address.java***, all text files and all unit test files