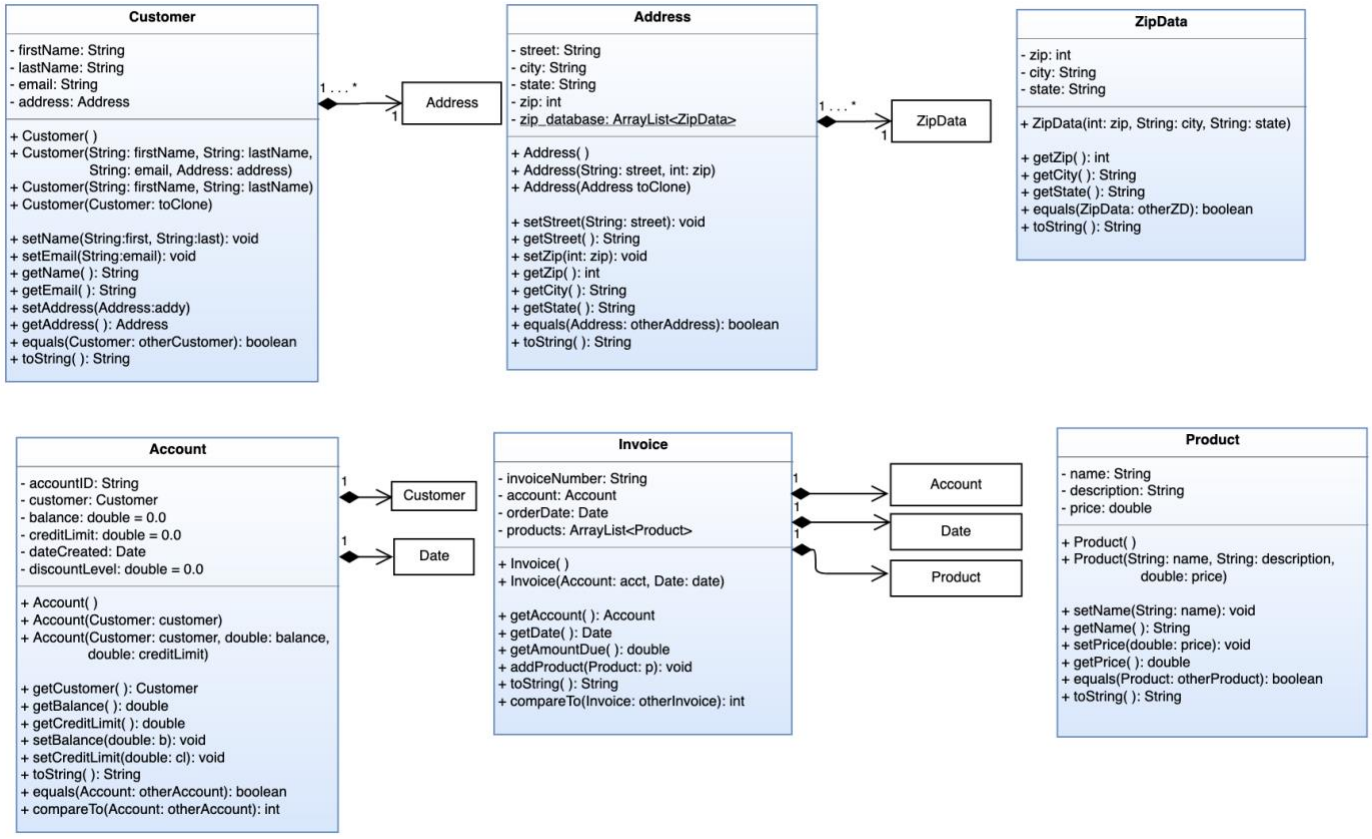
CSCI165 Computer Science II
Object Oriented Composition
Programming Assignment

Building on the Discussion and Lab work from this module, your task in this assignment is to finish the Customer project by adding a couple of new classes. Here is the complete UML. Notice the addition of the Invoice and Product class.

Privacy must be protected!!



Invoice: An invoice shows that a certain number of Products were charged to an Account. Include the following

- A Date object representing the ***date of purchase***. This can be set to any date
- An Account object representing the ***customer account responsible for the order***
- **An invoice number:** Take the customer first and last name and concatenate them together along with the date with slashes removed.
    - **Example:** Ken Whitener 12/12/2020
    - **Invoice Number:** KenWhitener12122020
- An ArrayList of the Products that are on the Invoice.

- An **addProduct** method that allows for the addition of a ***single Product*** to the products list.
- The method **getAmountDue** will iterate through the Products list and create a sum of all the prices. This amount should be included in the toString
- The **compareTo** method should be defined around the amount due.

Product: Design the Product class according to the UML diagram above.

Unit Tests: Write unit tests for the following ***Invoice methods***

- compareTo
- getAmountDue

Application: Ask questions if anything is unclear. Create a Driver class that satisfies the following

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. These data are tab separated.
- The last field in the rows is an optional ***sku number*** you may ignore this, or you can add a field for it in the Product class.

Customers, Address and Accounts: Create the Customer accounts identically to the lab. Copy and paste if you'd like.

Invoice: Create an array of 100 Invoice objects

- Randomly select accounts from the accounts array mentioned above. Accounts can be duplicated
- For each Invoice randomly add between 1 and 20 products by calling ***addProduct*** for each one

When the Invoices have been created, iterate through the array calling toString on each instance. Show one Invoice at a time and require a button press to show the next one. The amount due should be included in the output.

Submit: push ***Invoice.java, Customer.java. Product.java, Account.java, Address.java***, all text files and all unit test files