DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF NEBRASKA—LINCOLN

Order Invoice System

CSCE 156 – Computer Science II Project

Student 003

4/3/2013

Version 5.0

The following is a document detailing the design and implementation of an updated order invoice system for Jenn-Eric Import/Export Company.

Revision History

Version	Description of Change(s)	Author(s)	Date
1.0	Initial draft of this design document	**Student 003**	2013/01/31
2.0	Updates to Phase 2 of Project	**Student 003**	2013/02/12
2.1	Addition of UML	**Student 003**	2013/02/13
3.0	Database Design	**Student 003**	2013/02/28
4.0	JDBC Interface	**Student 003**	2013/03/12
5.0	Data Structure Design and Integration	**Student 003**	2013/04/03

Contents

1. Introduction

This document covers the development, testing, and implementation of an order invoice system developed using Java which manages the customers, products and orders of the Jenn-Eric Import/Export Company. The customers, products, and orders are organized into tables in a database. The tables of data in the database are instantiated by the java system and rows of data are entered through the JDBC.

1.1 Purpose of this Document

The purpose of this document is to describe design techniques, including failures and successes, testing methods, and implementation strategies over the timeline of the project being covered.

1.2 Scope of the Project

The Java based invoice report system created for the Jenn-Eric Import/Export Company is designed to manage orders made by the various customers of the company and prepare and output detailed individual order invoices as well as an overall summary report of orders made to the company. The system organizes the products offered by the company, the customers of the company and the orders that are placed by customers into classes of data with methods that act on that data.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

More information provided in subsequent revision.

1.3.2 Abbreviations & Acronyms

XML – Extensible Markup Language
UML – Unified Modeling Language
OOP – Object Oriented Programming
JDBC – Java Database Connectivity
JPA – Java Persistence API
API – Application Programming Interface
ADT – Abstract Data Type

2. Overall Design Description

As discussed in the project scope, there are three entities that the application supports: products, customers, and orders.

Each product has a unique alpha-numeric identification number, a name, a description, and a current per-unit price. In addition, each product is organized under a category.

Each order that is placed through the company is defined by: a unique alpha-numeric ID, the date it was placed, the products in the order and their quantities, and the customer who placed the order.

Each customer that makes orders through the company falls under one of three categories: Personal, Corporate, or Academic. All customers are defined by: an alpha-numeric ID, customer type, a unique address, and 0 or more email addresses. Personal customers have a first and last name. Corporate and academic customers have a business name. Each type of customers' order is processed differently based on tax rates, special fees, and order discounts. The tax rates, special fees, and discount rates are:

- Personal order:
    - Sales tax = 7%
    - Service fees = none
    - Discounts
        - Product purchase >= 10 receives 10% discount
        - Per-order discount = none
- Academic order:
    - Sales tax = none
    - Service fees = none
    - Discounts
        - 10 <= Product Purchase < 20 receives 10% discount
        - 20 <= Product Purchase receives 20% discount
        - Per-order discount = 5%
- Corporate order:
    - Sales tax = 3%
    - Service fees = $100 per order
    - Discounts
        - 10 < Product Purchase < 50 receives 10% discount
        - 50 <= Product Purchase < 100 receives 15% discount
        - Product purchase >= 100 receives 20% discount
        - Per-order Discounts
            - $1000 <= Pre-discount Total < $5000 receives 10% discount
            - Pre-discount Total >= $5000 receives 20% discount

Orders are processed in the following manner. A pre-discount total is caluclated based on the unit price and quantity of each product. A per-item bulk discount rate is then applied. A per-order discount rate is then applied. Services fees are then added to the total to give a subtotal. The grand total is then calculated from the subtotal and the tax rate of the customer. At every stage, each subtotal and discount is rounded to the nearest cent.

The application that is developed models this "business logic" and produce both summary reports and detailed order invoices.

3. Detailed Component Description

The application design uses some important OOP concepts to create a robust and easily maintainable code design. Each class utilizes the OOP concept of encapsulation keeping the attributes of that class

private. The concept of inheritance is applied in the Customer class which has three subclasses: Person, Corporation, and Institution. Inheritance is also used in the Order class which has three subclasses: PersonalOrder, CorporateOrder, InstitutionalOrder. This provides the ability to use instances of the subclasses in place of the superclass which is an advantageous strategy in OOP.

3.1 Class/Entity Model

The following two pages show a UML (Figure 1) covering the classes that make up the application and how each application interacts with the others and an EER diagram (Figure 2) showing the tables in the database and how they are connected via primary and foreign keys.

Invoice Report System UML

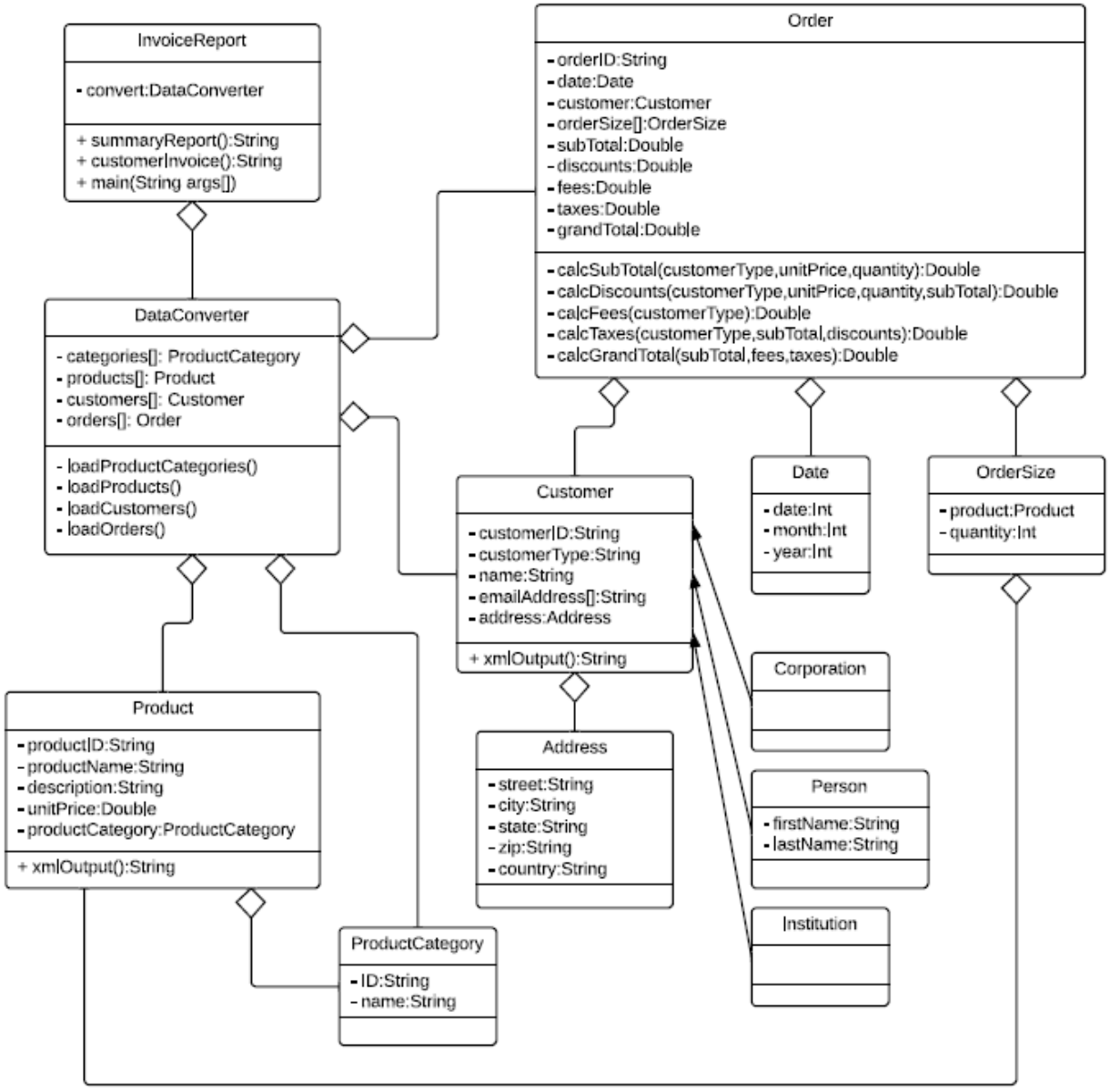

Figure 1: **UML Diagram**

ER Diagram of Database Design

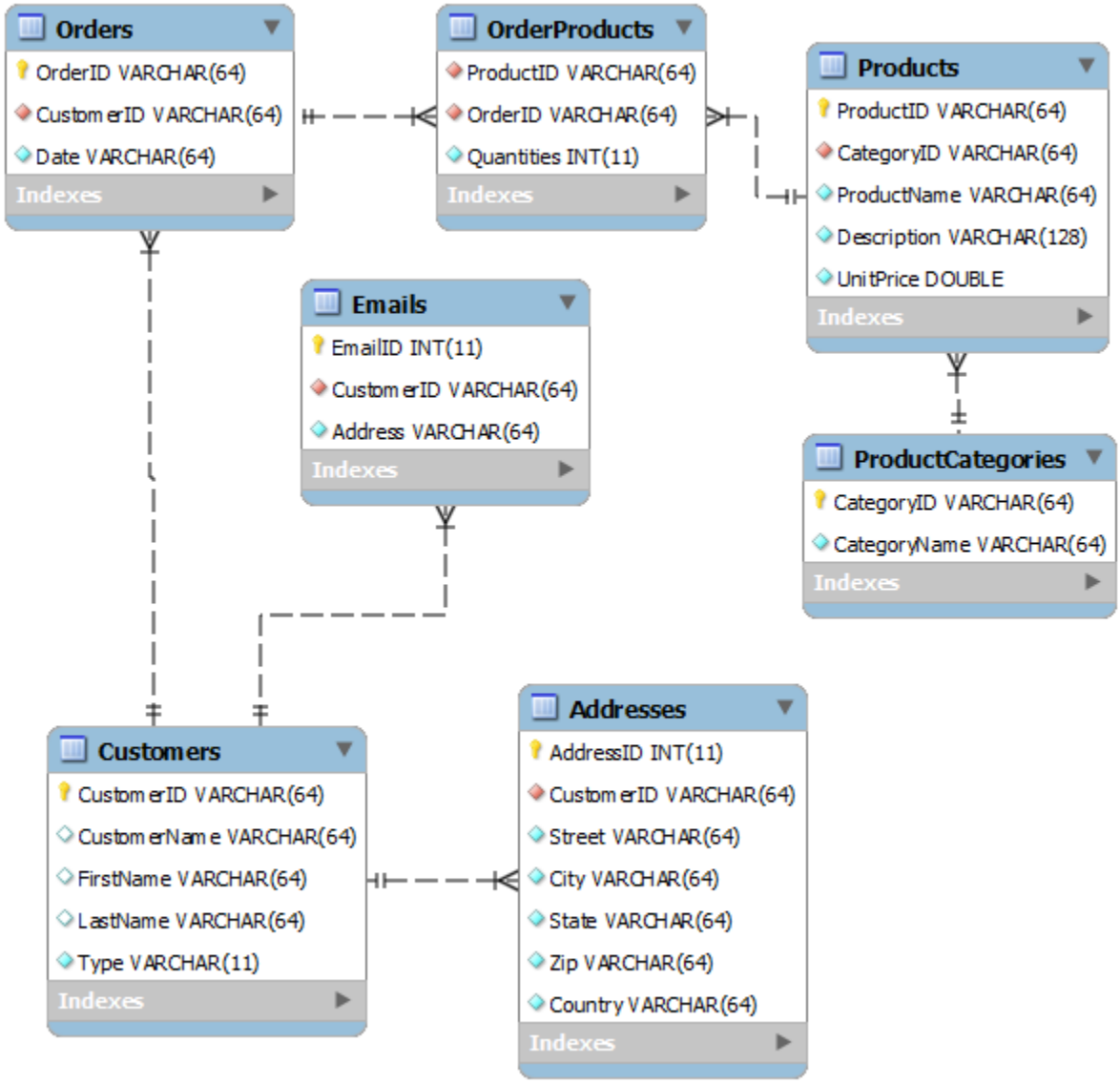

Figure 2: **ER Database Design**

3.1.1 Java Application Testing Strategy

The test case used to verify the system provides the system with Customers.dat, ProductCategories.dat, Products.dat, and Orders.dat. These are four input files containing data the system acts on to generate Summary Reports and Customer Invoices. The data is read in and organized properly into arrays of objects for all four inputs. Calculations take place in the Order class for each order and the main function formats the output into Summary Reports and Customer Invoices. The results are visually verified.

3.1.2 Database Testing Strategy

Testing consists of using the following queries and what happens when they are used:

Query	Output
Add existing table	Error: cannot add table that already exists
Add customer without ID, Type, Name or Address	Error: ID, Type, Name or Address not allowed to be NULL
Add product without ID, Name, Description, UnitPrice or CategoryID	Error: ID, Name, Description, UnitPrice or CategoryID not allowed to be NULL
Add product category without ID or Name	Error: ID or Name not allowed to be NULL
Add order without OrderID, CustomerID, Date or Products and Quantities	Error: OrderID, CustomerID, Date or Products and Quantities not allowed to be NULL

As well as the following :

1. Retrieve all * information
2. Retrieve a * based on a data point
3. Add a * to * table
4. Change a data point of a specific *
5. Remove a * record
6. Remove all * records

Where * is Customer, Order, Address, Product Category, Order Product

3.2 JDBC Operation

An API serves to load and persist data from the database in to the java application via JDBC. This API provides the appropriate methods to load and persist data to the database. Then a user interface layer uses the API.

A class called OrderData holds all methods necessary for interacting with the database via the JDBC. The methods include those to: remove all product categories, remove all products, remove all customers, remove customer, remove order, add product category, add product, add customer, add customer email, add order, add order product. Each of these methods sends a query to the database to complete the function of the method. This is how the API persists data to the database.

The methods described previously and detailed in the UML diagram are repurposed. Rather than reading from flat data files, they send queries to the database to populate array lists from the data in each table of the database. Each time a method from OrderData is called, the associated method is called to repopulate the array list pertaining to the table that was affected in the database.

In the main method of the java application, java lists in the application are populated with data from the database and used to calculate the pertinent information that is printed out on the standard output. These data points include individual order summaries as well as an overall invoice report.

Data validation occurs in the java program. When the user attempts to enter any product category, product, customer, or order that already exists an error is printed on the standard output. The data is not added to the table. The user is expected to enter valid data types in the API. When the user enters null data an exception is thrown.

3.2.1 JDBC Testing Methods

Attempting to enter invalid data types into the database is handled. The java application handles these by printing to the standard output that it was an invalid type and not attempting to enter it into the database.

3.3 Design & Integration of Data Structures

A sorted list ADT is designed and implemented to hold an arbitrary number of customers which maintains the order of the customers based on a data point. There are implementations to support three different orderings of data which are: based on customer name, first on customer type (personal, academic, corporate) and then based on name(company name for academic and corporate, last name for personal), based on grand total of all orders highest-to-lowest, based on customer type then alphabetically based on customer ID. These entries are sorted alphabetically in descending order. The program still connects to the database that was created.

Linked lists are used for the ADT classes in the application. The ADT methods utilized facilitate adding, removing, and retrieving/iterating elements. Order is maintained rather than updated or manipulated.

When data is added to a linked list a method iterates through the current list to find the location that the new data should be placed. The preceding and proceeding link information will be held by local objects, then the preceding link will link to the new data and the new data will link to the proceeding link.

When data is removed from the linked list the preceding and proceeding link information will be held by local objects, the data to be removed is deleted then the preceding link will link to the proceeding link.

3.3.1 Component Testing Strategy

Testing methods include: adding and removing links from the list then printing the information to the standard output to verify that the linked list was updated properly.

3.4 Changes & Refactoring

The methods described in section 3.1 and detailed in the UML diagram are repurposed. Rather than reading from flat data files, they send queries to the database to populate array lists from the data in each table of the database.

4. Bibliography

[1] Eckel, B. (2006). *Thinking in Java* (4th ed.). Prentice Hall.

[2] *Lucid Chart.* (2012). Retrieved February 13, 2013, from http://www.lucidchart.com