DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF NEBRASKA—LINCOLN

FarMarT Business Operations

Matthew Schlatter

3/3/2023

3.0

This document outlines a sales subsystem for a farming company, FarMarT.

Revision History

This table documents the various major changes to this document.

Version	Description of Change(s)	Author(s)	Date
1.0	Initial draft of this design document	Matthew Schlatter	2023/02/17
2.0	Updated to be more technical	Matthew Schlatter	2023/02/24
3.0	Updated UML chart and added transition to the chart	Matthew Schlatter	2023/03/03

Contents

Revision History 1
1. Introduction 4
1.1 Purpose of this Document 4
1.2 Scope of the Project 4
1.3 Definitions, Acronyms, Abbreviations 4
1.3.1 Definitions 4
1.3.2 Abbreviations & Acronyms 4
2. Overall Design Description 4
2.1 Alternative Design Options 5
3. Detailed Component Description 5
3.1 Database Design 5
3.1.1 Component Testing Strategy 5
3.2 Class/Entity Model 6
3.2.1 Component Testing Strategy 7
3.3 Database Interface 7
3.3.1 Component Testing Strategy 8
3.4 Design & Integration of Data Structures 8
3.4.1 Component Testing Strategy 8
3.5 Changes & Refactoring 8
4. Additional Material 8
5. Bibliography 8

1. Introduction

This design creates a sales subsystem for the family-owned business FarMarT (FMT), a regional chain of B2B (business-to-business) stores that sell farm equipment and supplies to farmers large and small, such as seeds, tractors, soil, etc. With the recent acquisition by Pablo Myers, a revamp of this system is favorable to increase profits. This entire system is independently developed, and the sales subsystem efficiently gathers and sorts valuable data on customers, customer orders, stores, and invoices. Totals are calculated with any applicable taxes plus the price of the customer's desired item and amount. For example, if the customer would like 10 bags of soil, at $10/bag, this would cost $107.15 with tax because products have a tax of 7.15%. Furthermore, services have a tax of 3.45%, purchased equipment or leases less than $10000 have no tax, leases $10000-$100000 have a $500 tax, and leases >= $100000 have a tax of $1500.

1.1 Purpose of this Document

This document is designed to document the process of creating this subsystem. It lays an outline for the entire project and explains chosen design decisions, as well as pros and cons of many design options.

1.2 Scope of the Project

This project is designed to be utilized by FarMarT, specifically following their business model and procedures. It is designed to make orders easily accessible and trackable. This subsystem efficiently collects data and stores it in a database using Electronic Data Interchange where it can be gathered to create reports. This system does not deal with customer issues or marketing, only the tracking of sales.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

All necessary definitions for this document can be found here.

1.3.2 Abbreviations & Acronyms

All necessary abbreviations and acronyms for this document can be found here.

FMT – FarMarT (Company Name)

2. Overall Design Description

This project takes in many different data points and provides different data files based on the provided information. Data files include invoices with customer orders and order information, customer data including name and contact information, as well as store information where the order was placed. Person, store, and item data is imported through CSV files and exported as XML and/or JSON files. Invoice data is printed to the standard output in the format of a receipt. Additionally, a SQL database is implemented to make the management of mass amount of data from these orders easier.

2.1 Alternative Design Options

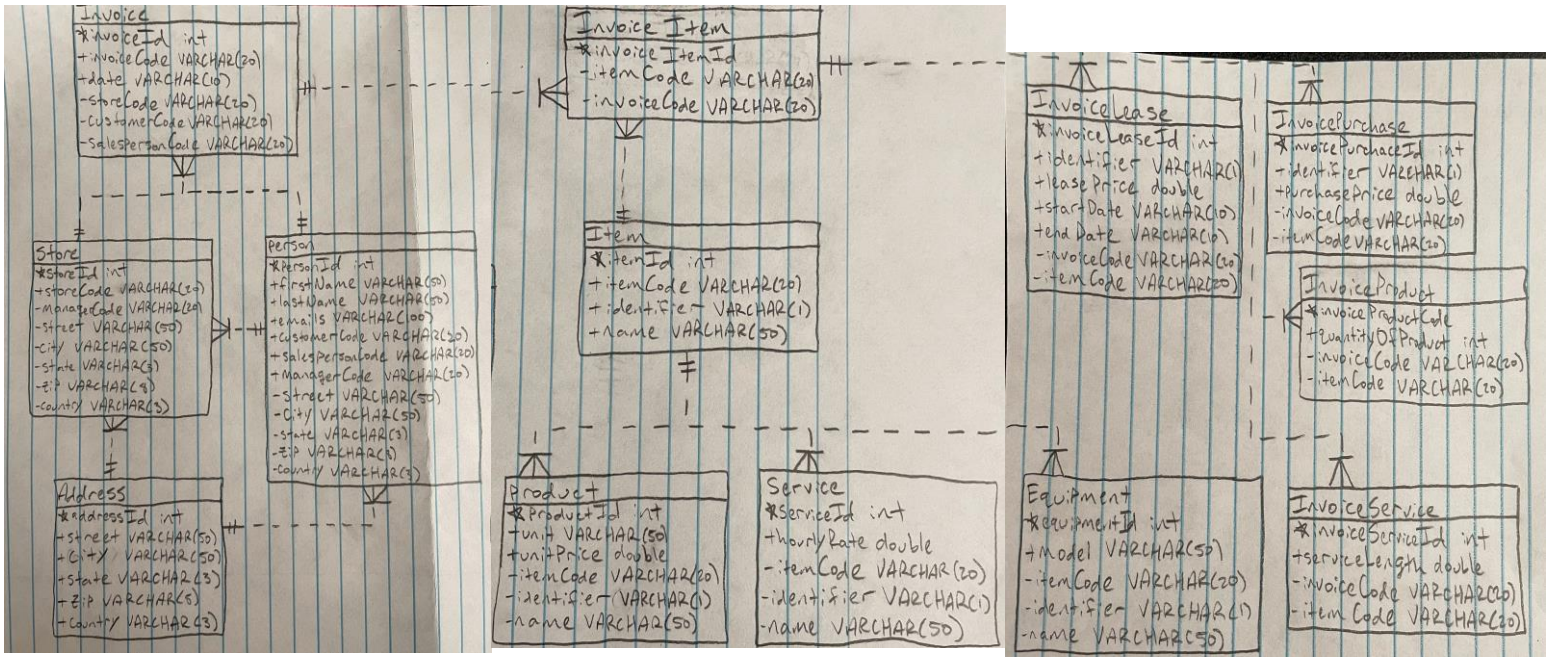
Loading and converting data can be placed into a separate package to allow further separation of operations. Additionally, interfaces can be implemented to increase inheritance and reduce issues with abstract classes.

3. Detailed Component Description

This design is based in Java with a connection to a SQL database.

3.1 Database Design

This database utilizes 11 tables to store data of different data types. Primary keys are denoted by a *, non-primary keys are denoted by a +, and foreign keys are denoted by a -. All foreign keys directly correlate to the key in the subsequent table and no value can be null. By utilizing a database, it will be simpler to deal with the massive amount of input data that will be received with orders as well as data corresponding to products and services.



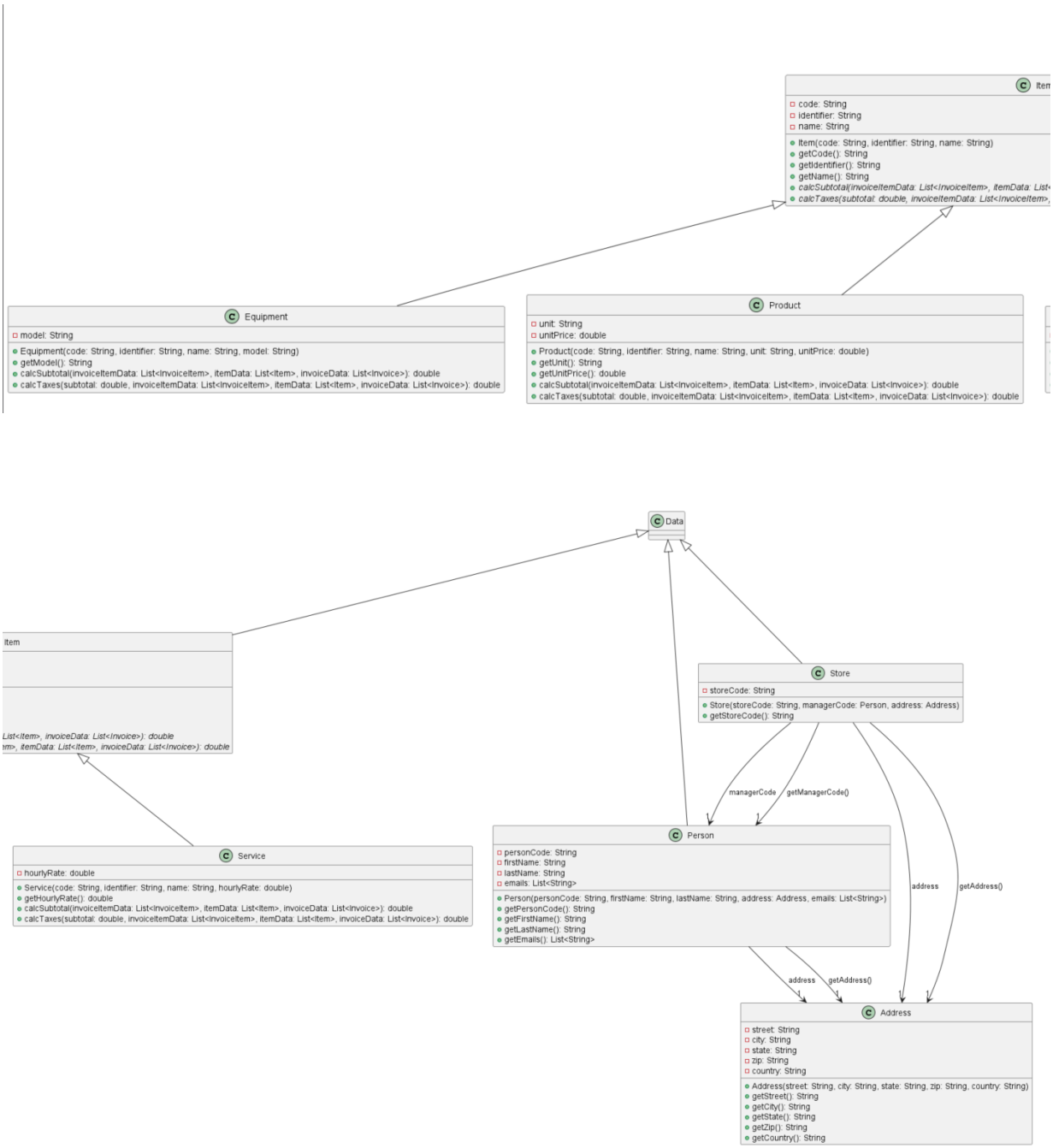
3.1.1 Component Testing Strategy

Data is inserted into the database and tested using multiple different queries to ensure the desired results are output. Queries include retrieving basic person or order information, adding emails to a record, finding specific item information, and total sales by employees or stores.

3.2 Class/Entity Model

Outline of classes and their specific variables and methods. There are three main sections. The first deals with all basic data types, the second deals with loading, converting, and outputting the data, and the last deals with invoices and invoice items.

(DISCLAIMER: this will be reworked for an assignment 3 resubmission)



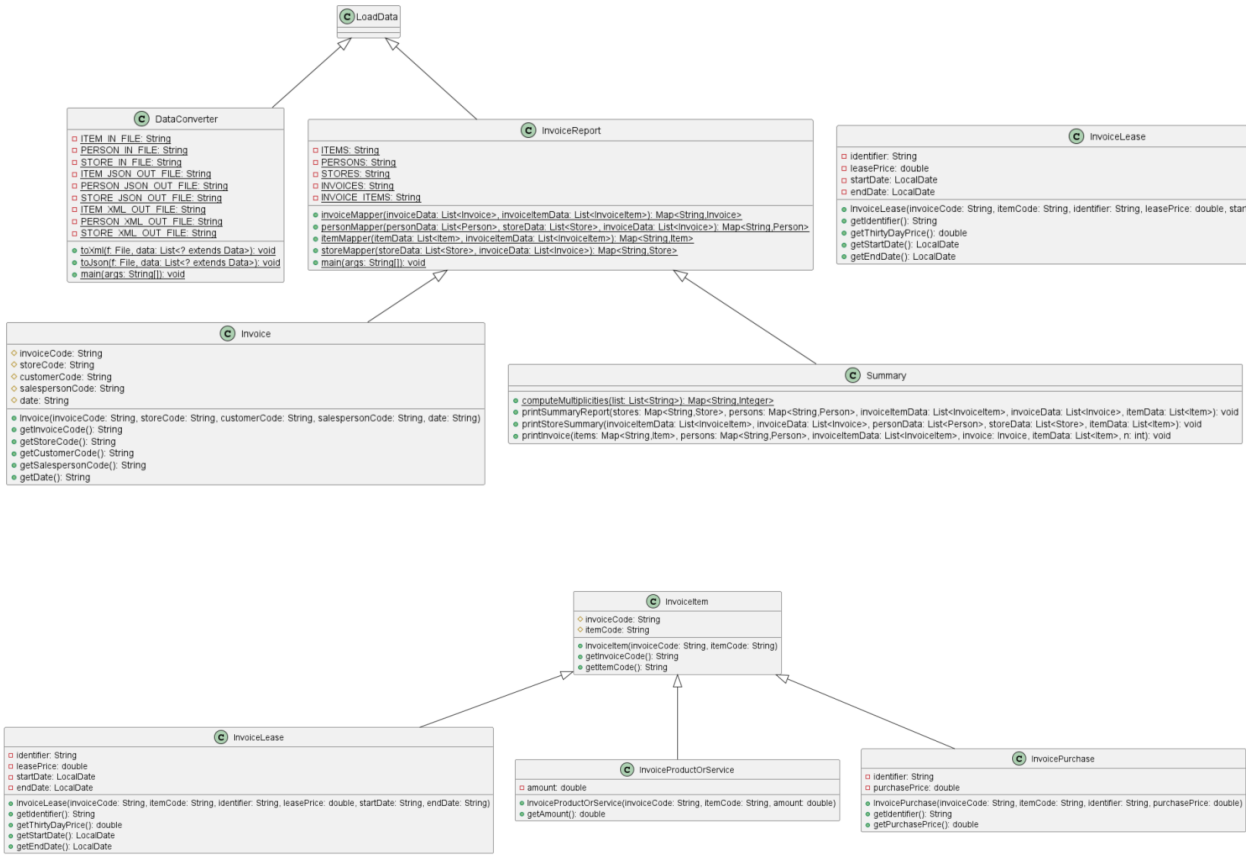


Figure 1: UML Diagram for the FarMarT operations, outlining different classes and variables.

3.2.1 Component Testing Strategy

Test files with randomly generated data from Mockaroo were input into the classes above to test the effectiveness of scanning and printing data. Data was imported through CSV files and exported to XML/JSON files. Each class passed their respective tests.

Additionally, test files for invoices were generated with Mockaroo data as well as utilizing the item, person, and store data. An output.txt file shows expected output of invoices in a receipt format. Totals were calculated by hand adhering to the business logic outlined in the Introduction of this document.

3.3 Database Interface

[This section will be used to detail phase IV where you modify your application to read from a database rather than from flat files. This section will detail the API that you designed—how it conformed to the requirements, how it worked, other tools or methods that you designed to assist, how it handles corner cases and the expectations or restrictions that you've placed on the user of the API. In earlier phases this section may be omitted or a short note indicating that details will be provided in a subsequent revision of this document. See the full rubric for what belongs in the introduction!]

Table 1: Average Performance on Assignments; on-time vs. late and individual vs partners. In general, captions for Tables should appear *above* the table.

	1	2	3	4	5	6	7
On-time	93.16%	88.06%	87.89%	89.37%	83.42%	88.40%	74.56%

	(78.46%)	(72.31%)	(67.69%)	(56.92%)	(29.23%)	(53.85%)	(75.38%)
Late	88.75% (12.31%)	85.28% (20.00%)	70.32% (15.38%)	90.40% (15.38%)	82.74% (44.62%)	94.22% (15.38%)	N/A
Diff	4.42%	2.79%	**17.57%**	1.03%	0.68%	5.82%	-
Individual	NA	88.43% (73.85%)	82.32% (33.85%)	87.22% (27.69%)	86.40% (23.08%)	82.67% (26.15%)	
Pairs	NA	83.55% (18.46%)	86.22% (49.23%)	91.00% (46.15%)	78.53% (49.23%)	92.83% (46.15%)	
Diff	NA	4.88%	3.90%	3.78%	7.87%	10.16%	

3.3.1 Component Testing Strategy

[This section will describe your approach to testing this component. Describe any test cases, unit tests, or other testing components or artifacts that you developed for this component. What were the outcomes of the tests? Did the outcomes affect development or force a redesign?]

3.4 Design & Integration of Data Structures

[This section will be used to detail phase V where you design an original data structure and integrate it into your application. In earlier phases this section may be omitted or a short note indicating that details will be provided in a subsequent revision of this document? See the full rubric for what belongs in the introduction!]

3.4.1 Component Testing Strategy

[This section will describe your approach to testing this component. Describe any test cases, unit tests, or other testing components or artifacts that you developed for this component. What were the outcomes of the tests? Did the outcomes affect development or force a redesign?]

3.5 Changes & Refactoring

[During the development lifecycle, designs and implementations may need to change to respond to new requirements, fix bugs or other issues, or to improve earlier poor or ill-fitted designs. Over the course of this project such changes and refactoring of implementations (to make them more efficient, more convenient, etc.) should be documented in this section. If not applicable, this section may be omitted or kept as a placeholder with a short note indicating that no major changes or refactoring have been made.]

4. Additional Material

[This is an optional section in which you may place other materials that do not necessarily fit within the organization of the other sections.]

5. Bibliography

[This section will provide a bibliography of any materials, texts, or other resources that were cited or referenced by the project and/or this document. You *must* consistently use a standard citation style such as APA [1] or MLA.

[1] *APA 6 – Citing Online Sources*. (n.d.). Retrieved March 19, 2021, from https://media.easybib.com/guides/easybib-apa-web.pdf

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