|  |
| --- |
| Department of computer science & Engineering  University of Nebraska—Lincoln |
| Simple Invoice System |
| CSCE 156 – Computer Science II Project |
|  |
| **Wei Cao and Cameron Cunning** |
| **9/18/2014**  **[Version 1]** |

|  |
| --- |
| [Provide a descriptive abstract here—a short summary of this document and the project that it documents.] |

# 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 | Cao/ Cunning | 2014/09/18 |
|  |  |  |  |

Contents

[Revision History 1](#_Toc349390336)

[1. Introduction 4](#_Toc349390337)

[1.1 Purpose of this Document 4](#_Toc349390338)

[1.2 Scope of the Project 4](#_Toc349390339)

[1.3 Definitions, Acronyms, Abbreviations 4](#_Toc349390340)

[1.3.1 Definitions 4](#_Toc349390341)

[1.3.2 Abbreviations & Acronyms 4](#_Toc349390342)

[2. Overall Design Description 4](#_Toc349390343)

[2.1 Alternative Design Options 4](#_Toc349390344)

[3. Detailed Component Description 4](#_Toc349390345)

[3.1 Class/Entity Model 4](#_Toc349390346)

[3.1.1 Component Testing Strategy 5](#_Toc349390347)

[3.2 Class/Entity Model 5](#_Toc349390348)

[3.2.1 Component Testing Strategy 5](#_Toc349390349)

[3.3 Database Interface 5](#_Toc349390350)

[3.3.1 Component Testing Strategy 5](#_Toc349390351)

[3.4 Design & Integration of Data Structures 5](#_Toc349390352)

[3.4.1 Component Testing Strategy 5](#_Toc349390353)

[3.5 Changes & Refactoring 5](#_Toc349390354)

[4. Additional Material 5](#_Toc349390355)

[5. Bibliography 6](#_Toc349390356)

# Introduction

This is the Software Design Description for the Simple Invoice System designed for Cinco Computer Consultants(CCC). It outlines the technical design of the application that is being developed to provide services to regional companies including equipment and services such as training, consultations and licenses.

## Purpose of this Document

The purpose of this document is to describe the design techniques of the new system. It covers failures and successes, testing methodology, and implementation strategies. This document also outlines the MySQL database and its relationship to the newly developed Java application. With the document, you can see clearly both the database model and the class model. You also have seen that all parts of the program have been fully tested.

## Scope of the Project

The Java based simple invoice system developed for Cinco Computer Consultants(CCC) is designed to replace their aging AS400 green-screen system. It is designed to manage portfolios of various products for their customers. These products include equipment, consultations and licenses. It can produce the summary detail for all invoices. What more, it also can produce detail report for each invoice, including sub-total, fees and taxes for each invoice.

## Definitions, Acronyms, Abbreviations

### Definitions

Equipment - various computer and electronic products that CCC sells to its clients as a re-seller.

Consultations - services that are offered to clients by CCC such as training, system evaluations, etc.

License - product that include things like software, server hosting, or third-party services.

Invoice - individual sales for products.

### Abbreviations & Acronyms

ADT - Abstract Data Type

ER Diagram - Entity Relationship Diagram

JDBC - Java Database Connectivity

OOP - Object Oriented Programming

SQL - Structured Query Language

UML Diagram - Unified Modeling Language Diagram

XML - Extensible Markup Language

# Overall Design Description

The application will follow an object oriented design pattern. Utilizing this pattern will ensure maintainability and extendibility through the application’s period of use. The current primary classes include: Person, Customer, Product, Equipment, Licenses, and Consultations.

The product class is made super class and from it three product subclasses are created. These subclasses include Equipment, Consultation and license. The different subclasses represent the different types of products offered.

The Person class is the superclass, and the Customer class is an extension of it.

The overall goal of the development of these classes is to promote reusability. This goal is reached via the implementation of the classes and subclasses outlined above.

## Alternative Design Options

Alternative design options considered in the development of this application:

1. Make the person class abstract.
2. Make the class Product abstract.
3. Make a class specifically for the handling of data conversion.

# Detailed Component Description

[Provide an introduction to this section here]

## Database Design

[This section will be used to detail your database schema design (Phase III). 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.]

### Component Testing Strategy

[This section will describe your approach to testing this particular component.]

## Class/Entity Model

[This section should detail your Java classes—their state, interface and how they relate to each other. It is highly recommended that you document these elements using tables, UML diagrams, and other visually-informative methods. Figures and tables should have proper captions and be referenced in the main text just like in Figure 1. You should provide subsections to organize your presentation as applicable.]



Figure : A UAV (Unmanned Aerial Vehicle) soars above Memorial Stadium. Figures should be numbered and properly captioned.

### Component Testing Strategy

[This section will describe your approach to testing this particular 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?]

## 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.]

Table : 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% (78.46%) | 88.06% (72.31%) | 87.89% (67.69%) | 89.37% (56.92%) | 83.42% (29.23%) | 88.40%  (53.85%) | 74.56%  (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% |  |

### Component Testing Strategy

[This section will describe your approach to testing this particular 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?]

## 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?]

### Component Testing Strategy

[This section will describe your approach to testing this particular 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?]

## 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.]

# 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.]

# 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 or MLA (good reference: <http://www.cws.illinois.edu/workshop/writers/citation/)>.]

[1] *Citation Styles*. (n.d.). Retrieved December 19, 2012, from [http://www.cws.illinois.edu/workshop/writers/citation/](http://www.cws.illinois.edu/workshop/writers/citation/))

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