|  |
| --- |
| Department of computer science & Engineering  University of Nebraska—Lincoln |
| S.C.H.Y.C Fitness Invoice System |
| CSCE 156 – Computer Science II Project |
|  |
| **Taylor Schrader, Logan McCarthy** |
| **10/27/1998**  **Version 1.0(Pre-Aplha)** |

|  |
| --- |
| The contents of this document layout the object-oriented software design for the new invoice system for S.C.H.Y.C Fitness |

# 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 | Taylor S , Logan M | 1998/10/27 |
|  |  |  |  |

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 software is designed for S.C.H.Y.C Fitness to replace their old Legacy invoice system which is based on flat flies with a new simple invoice system. This system will record old and new customer information.

## Purpose of this Document

The purpose of this document is to show how this program was designed and created using object oriented programming in java for the S.C.H.Y.C Fitness group.

## Scope of the Project

This java based invoice management system will allow S.C.H.Y.C Fitness group to keep track of and manage the services sold to customers. Services include Memberships, daily and yearly, parking passes, and equipment rentals.

[Describe the scope of the project, what features and functionality it covers (at a high-level). Describe the problem statement and context in which this project is being developed. Who is it for, what is it for, etc.? You may also explicitly indicate what is *not* within the scope—other potential pieces of the overall project that are not covered by this document]

## Definitions, Acronyms, Abbreviations

### Definitions

Object Oriented Programming - refers to a type of computer programming in which programmers define not only the data type of a data structure, but also the types of operations that can be applied to the data structure.

Flat File System- A file with no structure or indexing

### Abbreviations & Acronyms

[Define all abbreviations and acronyms used in this document here. This relieves you of the need to define such things within the context of the document itself and provides an easy reference for the reader.]

OOP – Object oriented programing

JSON – JavaScript object notation

XML – Extensible market language

JAR- Java archive

# Overall Design Description

To simplify the invoice system we will use a class base structure. Primary classes will include, Person, Membership type, Contact information

[Provide an overall summary/description of the project. Identify the major design components, technologies, etc.]

## Alternative Design Options

No alterative design options have arisen yet.

[If applicable, describe and discuss alternative design options that you considered and discuss why they were not chosen. What advantages and disadvantages do the alternatives provide and what advantage/disadvantages do the chosen design elements provide. Provide some justification for why the chosen elements’ advantages/disadvantages outweighed the alternatives]

# Detailed Component Description

The program breaks down the different aspects of the invoice system into individual classes.

## 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

UML documents coming soon.

[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

Used the data from the flat file invoice system to test the transfer to XML and JSON.

[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

Pending phase IV documentation

[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

Pending phase V documentation

[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

Phase I – Initial design documentation.

[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

[1] Beal, Vangie. “OOP – Object Oriented Programming.” *Object-Oriented Programming Definition | Webopedia*, www.webopedia.com/TERM/O/object\_oriented\_programming\_OOP.html.