**System Project Abstract:**

Design a system of your choosing and understanding based on OOP and design in Java that adheres to the project requirements below. You should pick a project topic you are familiar with as you need to understand the problem to code. You will have little time to research. The project must be YOUR code. Any code not yours must be referenced via @see in your javadoc. You are free to observe systems to better understand how to code yours. When complete, we will analyze for design patterns to be implemented based on bad design structures in your code. The system should have at a minimum the following features:

**AddRecord:**  
For this feature add a record into the system based on your understanding of fields for the system. You should show persistence in this abstract (list file type and what record will look like).

**Show/ListData:**  
With the information provided in add record, the system shows a record list corresponding data by field content.

**ShowItemDetails**:

This feature allows users to see the items and the corresponding details given for the item while adding the item.

**SortRecords:**  
Shows the total records in the file and requests a field sortable feature based on your content in the project.

**SearchRecord:**  
When the function for this feature is first executed a request for a searchable field is presented. Results are presented based on single field search. If the record searched for is not found, the system project displays the message – “Error in opening! File Not Found!!”

**Edit/UpdateRecord:**  
This works in similar manner to the Search feature. The user must request a record to be updated via field capability (e.g. a “record number”). The project shows all the data in that record and the user can enter any data to modify. We will only do single record edits. If the record to be edited for is not found, it displays the message – “Error in opening! File Not Found!!”

**DeleteRecord:**  
This works in similar manner to the Search feature. The user must request a record to be deelted via field capability (e.g. a “record number”). A secondary prompt should occur to ask if sure you want to delete prior to performing the delete. We will only do single record deletes.

**CheckAvailability**:

If your system requires availability search, this feature should checks its availability. This would be true of service based, rental or purchase systems.

**Reservation / Reservation Information**:

This feature is very simple; it includes if necessary, the ability to reserve a service (e.g. car /airplane /bus/train), seat number and the passenger’s name.) The seat number of the particular vehicle is reserved under the passenger’s name. (Think Uber)

**Time/Bill Reporting/Display Information**:

It shows a report of all the items rented, purchased, etc. as itemized in the billing system or data display. A “before / after” approach could be ideal in situations of edit and deleting records.

This project will be coded using a GUI. You should implement error handling or error inhibiting in all instances. Your project must have a persistence system (file based) and you need to demonstrate CRUD operations with it as described above. You will need a UML data diagram (listed in Appendix A) to designate how your system is arranged before you start. The diagram can change as your project expands but you must document those changes using revision numbers, bold and strikethrough. Its purpose is to prevent code creep and maintain scale of the project. You are required to have at minimum 500 records for performance verification of the operations above. This is system based project not a data based project. The tool you built in the first few weeks should aid you in building data to successfully complete the project.

The guidelines and principles in the book are to be implemented as they are learned and you will be graded on content and your understanding of these guidelines and principles.

Telephone Directory / Management System / Address Book (Customer rolodex)

Calendar Application (Similar to Outlook uses Calendar and Contacts)

Object Collection (e.g. Baseball Card, Movie, Video, Comic Book, etc)

Inventory System / POS System ((Stores) Hardware, Music, Medical, Department, Book, Library, Supermarket, etc)

Rental / Reservation System (Bus, Hotel, Car, Plane, Rail, Theatre, etc)

Personal Diary System

Attendance Management System (Gym, College, Student Attendance)

TimeCard / Payroll System / Staff Management

Banking System

Periodic Table of Elements

Grading:

UML Diagram for Project (first submission)

Questions:

What level of GUI will I use?

What interface will be used to perform CRUD operations?

What file types will I use to save my data?

What data structures will be used for search and sort?