Group 2: Ryan Evans, Steven Chen, George Huang
IT 306 001
Group Project

Phase I – Design and Analysis

Project Name: A WalMart inventory system. Recording ITEMS, PRICE, and QUANTITY SOLD.

Application Description:
The application will allow users to create an item and store to a text file. The item can be group into 3 categories (Clothing item, book, and toy). Each category will share similar characteristics (price, quantity, number of item sold, total number of item sold by category, and a type). We will provide a text file for you to read from.

- User must declare either employee or admin.
- You will need to create the admin or employee from the file we provide before you can log in as either on the first use.
- Admins are employees; therefore, they can log in the same way as employees.
- Employees who aren't admins can't login as an admin
- If credentials are incorrect, the user should be notified of the reason.

We will provide a text file as an example of an employee and an admin. **You must add at least one new employee and one new admin for the program.** For this, you can re-write the file we provide. This is the only time you are permitted to re-write a file. Remember the file needs to be used in the program to log in the system and perform tasks. Remember that you cannot add any users without logging in the system as an admin. So, there should not be an option to add any users without logging in the system first.

Object Description:

Employees: Employees have a first, last name, email address(This will be used for the log-in id), and a password. Employees can:

- View items.
- Sort items based on number of items sold first, then quantity next (only one sort). And store in ONE new text file.(Using a method of your choice. Remember complexities)
- Search for an item

Admins share the same traits employees do, but they can:

- Order items
    - Think about what happens when you order an item. What changes and what stays the same.
- Add new items
    - When adding a new item, do not rewrite the whole file, only add a new item.
- Delete items
    - Again, do not rewrite, only delete.

- Add new employee

Items cannot be created without specifying a type(Clothing item, Book, or Toy)
All **Items** will have:

- Type
- Cost
- Quantity(The total number in stock of that particular item)
- Number of items sold.

Clothing item will share the traits of items, but will also have:

- A brand
- A Color
- Quantity per size(S, M, L, XL)

Book will share the traits of items, but will also have:

- Genre
- Author
- Title
- Quantity per title

Toy will share the traits of items, but will also have:

- Recommended Age
- Name
- Quantity per name

Validation Requirements:

- The program should never end unless the user indicates they want to exit the program. Meaning, every menu should have an option to exit.
- **The menu should be done using buttons (the swing class in JOP). You should NOT have "enter 1 to do this."** Think logically and make the program appealing to users.
- Use proper validation. If it would makes sense to have something validated be sure you do it.
- Email address should model the one given in the file provided. Ex: name@gmu.edu
- Password must include an uppercase letter, a number, and a special character(!@#$?).
- You are free to use any data structure you know, but the program must work efficiently.
- You must use object oriented programming.
- Inheritance should be used.
- Validating password and email are used for creating new employees
- If an incorrect password or email is entered, the user should be re-prompted

Other ideas:

- Try to make use of concepts learned in IT 206(Abstraction, modular programming, interfaces, relationships, information hiding, try/catch, throw/declare)
- The problem can be solved many ways, but some ways are better than others. Do your best to make the program as efficient as possible.

- Try to limit the outputs using JOP. Unless specifically told to output something, using a text file should suffice.
- Inputs can be done through JOP if you desire, but you can use CMD if you would like.
- There should be AT LEAST six classes. You may decide that you need to use more objects to solve the program and that is fine.