CS 2336 PROJECT 2 – Inventory Management

Part 1 Due: 9/28 by 11:59 PM
Project Due: 10/05 by 11:59 PM

KEY ITEMS: Key items are marked in red. Failure to include or complete key items will incur additional deductions as noted beside the item.

Submission:

- **The file containing main must be named Main.java. (-5 points)**
- **The project files must be in a package named InventoryManagement. (-5 points)**
- All project deliverables are to be submitted in eLearning until further notice.
- Zip the contents of the src directory into a single zipped file
    - Make sure the zipped file has a .zip extension (not .tar, .rar, .7z, etc.) (-5 points)
- Programs must compile and run with Java SE 22 or 23.
- Each student is responsible for developing unit test cases to ensure the program works as expected.
- **Type your name and netID in the comments at the top of all files submitted.** (-5 points)

Objectives:

- Use inheritance/polymorphism to create base and child classes and methods
- Utilize multiple classes in the same program
- Perform standard input validation
- Implement a solution that uses polymorphism

Problem: A small electronics company has hired you to write an application to manage their inventory. The company requested a role-based access control (RBAC) to increase the security around using the new application. The company also requested that the application menu must be flexible enough to allow adding new menu items to the menu with minimal changes. This includes re-ordering the menu items and making changes to the description of a menu item without having to change the code.

Security: The company has suggested to start the application by asking the user for a username and password to ensure that the user is authorized to access the application. There are two types of users at this company, managers and employees. If managers log on to the application, they will see all options on the menu list. If employees log on to the application, they will see a limited set of options on the menu list. User information is stored in Users.dat file, which may or may not exist at the start of the program. A super user "admin" with password "admin" has already been hardcoded in the program to allow for the initial setup and the creation of other users. The Users.dat file contains the FirstName, LastName, Username (case insensitive), HashedPassword and a flag to indicate whether a user is a manager or not. The file is comma separated and it is formatted as follows:

```
Joe, Last, jlast, 58c536ed8facc2c2a293a18a48e3e120, true
Sam,, sone, 2c2a293a18a48e3e12058c536ed8facc, false
Jane, Best, jbest, 293a18a48e3e12052058c536ed8facc2c, false
```

Application Menu: The menu of the application is dynamically loaded and displayed to the user only after the user successfully logs on. The menu items will be loaded from file "MenuList.dat", which may or may not exist at the start of the application. If the file doesn't exist, the application should show at least an Exit menu item as default. The file will contain all menu items details, including the name of the command that will be executed when the menu item is selected. If a menu item is marked as restricted (Boolean flag), only managers can see that item. The file contains the following comma separated fields, Description, a Boolean flag to indicate if the option is restricted to managers only, and the name of the menu command that will be executed when the option is chosen. The order and option number of a menu item may change depending on how they are listed in the file. The Exit option will always be listed last and it will not be in the file. Below is a sample of how the MenuList.dat file looks like:

```
Add User, true, AddUserCommand
Delete User, true, DeleteUserCommand
Change Password, false, ChangePasswordCommand
Add New Product, true, AddProductCommand
```

*Note: The command name of each menu item must match the name of the class that you will create in the code (See `AddProductCommand` class in the code for example).

Inventory: The inventory consists of multiple products of type *`Product`* stored in class *`ProductCatalog`*. The *`ProductCatalog`* is responsible of all inventory operations that add, remove, find and update a product. When printing a product information, the product retail price should be calculated and displayed as well. Retail price = (cost + (margin * cost/100)). A list of functions has been added to this class in the provided code template. You must implement all listed functions. The inventory products will be saved in file Inventory.dat, which may or may not exist when the program first starts. The file will contain the product unique id (int), product name (string), cost (double), quantity (int) and margin (int, integer that represents margin percentage). The Inventory.dat file is comma separated and formatted as follows:

```
3424, Smart Watch, 20.45, 23, 80
65454, Flat Screen TV, 465.98, 15, 35
435, Computer Monitor, 123.54, 84, 43
```

Program Flow:

- Program starts in main() method
- Prompt user for username and password
- Authenticate user and maintain the logged-on user object
- Load inventory
- Load and create menu list
- Display menu list and prompt the user for option
- Execute selected option
- Keep displaying the menu until the user chooses to exit

Output Format:

```
Enter username: some username
Enter password: some password //Repeat prompts until user is authenticated OR show error and option to exit.

Invalid username or password!

Press enter to continue or “Exit” to exit:

Enter username: some username
Enter password: some password

Welcome Firstname LastName!

Inventory Management System Menu //This is the header of the MenuList

// The order and option number of a menu item may change depending on how they are listed in the MenuList.dat file. The Exit option will always be listed last and it will not be in the MenuList.dat file.

1- Add user
2- Remove user
3- Change password
4- Add new product
5- Update product information
6- Delete product
7- Display product information
8- Display inventory
9- Exit

Enter your selection: 7
Enter product name: sMaRt wAtCh

Id     Name         Cost         Quantity     Retail
-----------------------------------------------------
3424   Smart Watch  $20.45       23           $36.81

//Repeat the menu after each command is executed
```

Unit Testing: A unit test method is required to test each of the methods listed below. These methods will be used by the unit testing framework to test the accuracy of your code.

```
InventoryManagementSecurity.AuthenticateUser
InventoryManagementSecurity.AddNewUser
InventoryManagementSecurity.RemoveUser
InventoryManagementSecurity.ChangePassword
MenuList.AddMenuItem(MenuItem menuItem)
ProductCatalog.AddUpdateProduct(Product product)
ProductCatalog.RemoveProduct(int productId)
ProductCatalog.FindProduct(int productId)
ProductCatalog.PrintProductInformation(int productId)
ProductCatalog.PrintInventoryList()
```

Grading:

- Coding standards, style and comments **(10 Points)**
- Unit testing methods x 10, one for each of the methods mentioned above **(20 Points)**
- The rest of the grade will be broken down as follows:
  - `InventoryManagementSecurity.AuthenticateUser` **(5 Points)**
  - `InventoryManagementSecurity.AddNewUser` **(5 Points)**
  - `InventoryManagementSecurity.RemoveUser` **(5 Points)**
  - `InventoryManagementSecurity.ChangePassword` **(5 Points)**
  - `MenuList.AddMenuItem()` **(20 Points)** //This includes implementing all commands for the menu list
  - `ProductCatalog.AddUpdateProduct(Product product)` **(10 Points)**
  - `ProductCatalog.RemoveProduct(int productId)` **(5 Points)**
  - `ProductCatalog.FindProduct(int productId)` **(5 Points)**
  - `ProductCatalog.PrintProductInformation(int productId)` **(5 Points)**
  - `ProductCatalog.PrintInventoryList()` **(5 Points)**

Notes:

- The "InventoryManagement" Maven solution zip file has been provided to you in eLearning.
- All *.dat files are assumed to be local to the current executable.
- Do not change the signature of the methods/constructors in the template. If in doubt, please ask.
- Your program is expected to have basic input validation (Only input from user).
- You may use ArrayList in this project
- Part 1 deliverables: Unit Test methods + Login function (no add or remove user is required in part 1).
- Part 2 deliverables: Functioning program including the Unit Test methods.