List of Entity Methods

Project Title: Warehouse Internal Management System

By //TODO: Team Name

Elijah Rey Espiritu
Hoyeon Moon
Binkang Yu

California State University Long Beach

CECS 343

1. User

1.1. Method: getPassword

1.1.1. Description: Returns the user's password stored in the database.

1.1.2. Preconditions: User's password is already set and stored in the database.

1.1.3. Postconditions: Password of the user's account is returned to the control object .

1.1.4. Signature: String getPassword();

1.2. Method: setPassword

1.2.1. Description: Sets the authorization password for the user based on the inputted string

1.2.2. Preconditions: A user must have no password. The password must be non-empty between 5 to 15 characters.

1.2.3. Postconditions: The user's password is set in the database

1.2.4. Signature: void setPassword(String password);

1.3. Method: changePassword

1.3.1. Description: Changes the authorization password for the user based on the inputted password string.

1.3.2. Preconditions: Password must be non-empty and between 5 to 15 characters.

1.3.3. Postconditions: Password of the user account is updated in the database.

1.3.4. Signature: void changePassword(String password);

2. Customer

2.1. Method: addCustomer

2.1.1. Description: Add a new customer with their information based on user inputted info.

2.1.2. Precondition: The customer's information must not already exist in the database.

2.1.3. Postconditions: A new customer is created.

2.1.4. Signature: void addCustomer(String name, String status, String

orderHistory, double salesTax);

2.2. Method: setSalesTax

2.2.1. Description: Set a designated sales tax for a specific customer based on user input.
2.2.2. Precondition: At least one customer is in the database.
2.2.3. Postcondition: Sales tax is set for the customer.
2.2.4. Signature: double setSalesTax(double tax);

2.3. Method: changeStatus

2.3.1. Description: Mark a customer inactive/active based on the user's discretion. If the user inputs 1, the customer is marked active. If they input 0, the customer is marked active.
2.3.2. Precondition: At least one customer is created in the database
2.3.3. Postcondition: The customer has been marked inactive/active.
2.3.4. Signature: void changeStatus(int status);

2.4. Method: getAssociatedInvoices

2.4.1. Description: List all the orders the customer has made.
2.4.2. Precondition: At least one customer is in the database.
2.4.3. Postcondition: All the order history of this customer is listed.
2.4.4. Signature: String getAssoicatedInvoices();

3. Product

3.1. Method: addProduct

3.1.1. Description: Add a new product to the system based on user input. This does not increase the quantity of the product in any warehouse (use replenishStock below instead).
3.1.2. Preconditions: The item the user inputted does not exist in the selected warehouse.
3.1.3. Postconditions: The new product is added.
3.1.4. Signature: void addProduct(String productName, costPrice, sellingPrice, totalCost);

3.2 Method: ModifyProduct

3.2.1 Description: Modify a currently existing product.

3.2.2 Preconditions: The item the user is trying to modify exists in the database.

3.2.3 Postconditions: The user inputted product is modified

3.2.4 Signature: void modifyProduct();

4. Invoice

4.1. Method: openNewInvoice

4.1.1. Description: An invoice for a customer is created.

4.1.2. Preconditions: Customer must be new or has paid their last invoice.

4.1.3. Postconditions: New invoice for the customer is created.

3.1.4. Signature: void openNewInvoice(String invoiceNumber, String status, String shippingAddress, String deliveryMethod, double financeCharge, String orderDate, double deliveryCharge, double totalCharge);

4.2. Method: closeInvoice

4.2.1. Description: A customer invoice is closed.

4.2.2. Preconditions: The customer the user is trying to close an invoice for exists.

4.2.3. Postconditions: The customer's invoice is closed.

4.2.4. Signature: void closeInvoice();

4.3. Method: showOpenInvoices

4.3.1. Description: Shows all currently open invoices.

4.3.2. Preconditions: There is at least one open invoice in the database.

4.3.3. Postconditions: The currently open invoices are displayed.

4.3.4. Signature: void showOpenInvoices();

4.4. Method: showClosedInvoices

4.4.1. Description: Shows all currently closed invoices.

4.4.2 Preconditions: There is at least one closed invoice the database.

4.4.3. Postconditions: The currently closed invoices are displayed.

4.4.4. Signature: void ShowClosedInvoices();

4.7 Method: markShipped

4.7.1. Description: Marks the products in an invoice shipped

4.7.2. Preconditions: An unmarked invoice exists

4.7.3. Postconditions: Quantities in warehouse deducted by the amount on the invoice

4.7.4. Signature: void markShipped();

5. Salesperson

5.1. Method: addSalesperson

5.1.1. Description: User adds a salesperson to the system.

5.1.2. Preconditions: The salesperson does not already exist in the database.

5.1.3. Postconditions: A new salesperson is added to the system.

5.1.4. Signature: void addSalesperson(String firstName, String lastName, String gender, String dateOfBirth, double totalCommision, int totalSales, String startDate, double commissionRate);

5.2. Method: setCommissionRate

5.2.1. Description: User set commission rate for the salesperson.

5.2.2. Preconditions: At least one salesperson is working.

5.2.3. Postconditions: A new commission rate is applied to the salesperson.

5.2.4. Signature: void setCommissionRate(double commissionRate)

5.3 Method: showAllPerformances

5.3.1. Description: Shows all salesperson's performance.

5.3.2. Preconditions: There is at least one salesperson.

5.3.3. Postconditions: Displays total amount of sales and total commission.

5.3.4. Signature: void showAllPerformances()

5.4 Method: payCommission

5.4.1. Description: User picks a salesperson to pay commission from a list.

5.4.2. Preconditions: There is at least one salesperson in the database.

5.4.3. Postconditions: The salesperson is paid based on their rate.

5.4.4. Signature: void payCommission()

6. Warehouse

6.1. Method: addWarehouse

6.1.1. Description: User adds a new warehouse that can store items.

6.1.2. Preconditions: The warehouse does not already exist in the database.

6.1.3. Postconditions: A new warehouse is added to the database.

6.1.4. Signature: void addWarehouse(String name, String address, String city, String state, String zip, String phoneNumber)

6.2 Method: showProductUnderFive

6.2.1. Description: Displays the products that have <= 5 quantity by warehouse.

6.2.2. Preconditions: There is at least 1 item in a warehouse

6.2.3. Postconditions: The products are displayed in increasing order by quantity on hand.

6.2.4. Signature: void showProductsUnderFive()

6.3 Method: replenishStock

6.3.1 Description: Replenish the quantity for an item based on user input.

6.3.2 Preconditions: The item the user wants to replenish exists in the database.

6.3.3 Postconditions: The product's stock is replenished.

6.3.4 Signature: void replenishStock(String product, int quantity);

6.4 Method: displayAllProducts

6.4.1. Description: Display all products by warehouse.

6.4.2. Preconditions: There is at least one item in a warehouse.

6.4.3. Postconditions: The items are displayed in decreasing order of profit percent.

6.4.4. Signature: void displayAllProducts()

6.5. Method: getQuantity

6.5.1. Description: Show the current quantity for the product that the user selected.

6.5.2. Preconditions: The product the user is calling this method on exists in the database.

6.5.3. Postconditions: Quantity of the product the user selected is shown.

6.5.4. Signature: String getQuantity();