Tasks

Task List by Commit:

This is the sequence of tasks, grouped by commit, which follows the order of your features (High, Medium, Low) to logically build the system.

- **Task 1:**
  - **Feat:** Base project structure and (MVC) models
  - **Objective:** Create the project skeleton and all data classes.
  - **Classes to program:**
    - FinvoryApp: Create the main method.
    - FinvoryView: Create the class with a Scanner.
    - FinvoryController: Create the class and the constructor that receives data, view, db.
    - model/ (All 15 classes): Address, AdminDetail, CompanyAccount, Customer, Inventory, InventoryOfObsolete, InvoiceLineSim, InvoiceSim, PersonalAccount, Product, ProductSold, Report, ReturnedProduct, Supplier.
  - **Methods to program:** Only the attributes, constructors (empty and complete), and basic getters/setters for all model classes. No business logic is added yet.

- **Task 2:**
  - **Feat:** Implement hybrid persistence (JSON/CSV)
  - **Objective:** Create the logic for saving and loading data.
  - **Classes to program:**
    - Database: The main persistence class.
    - FinvoryData: The container class that will be saved as JSON.
  - **Methods to program:**

        - Database.load() and Database.save()
        - Database.loadDatabaseJson() and Database.saveDatabaseJson()
        - Database.loadCustomersCsv() and Database.saveCustomersCsv()
        - Database.loadSuppliersCsv() and Database.saveSuppliersCsv()
        - FinvoryApp.main(): Update to call db.load() and db.save().
        - FinvoryData: Mark customers and suppliers as transient.

- **Task 3**:
    - **Feat (high):** Implement core sales and inventory management
    - **Objective:** Implement High-Priority features 2, 4, and 8 (Sale, Obsolete, Location).
    - **Classes to program:**
        - FinvoryController: startMainMenu(), handleNewSale(), handleInventoryMenu(), handleProcessReturn(), handleManageObsolete(), findCustomer(), findProduct().
        - FinvoryView: showMainMenu(), showInventoryMenu(), askCustomerQuery(), askProductId(), askQuantity(), chooseInventoryToSellFrom(), askReturnReason(), askObsoleteAction().
        - model/Product: Refactor to remove stock.
        - model/Inventory: Implement productStock (HashMap), getStock(), removeStock().
        - model/InventoryOfObsolete: Implement productStock (HashMap), addStock(), removeStock().
        - model/ReturnedProduct: Implement the class.

- **Task 4:**
    - **Feat (high):** Implement revenue calculation and tax management
    - **Objective:** Implement High-Priority features 1 and 3 (Revenue and Taxes).
    - **Classes to program:**
        - model/FinvoryData: getTotalGrossProfile(), getTotalGrossDay(), setTaxRate(), getTaxRate().

        - model/InvoiceSim: calculateTotals(taxRate).
        - FinvoryView: showAdminMenu(), askNewTaxRate(), showDashboard().
        - FinvoryController: handleAdminMenu(), handleSetTaxRate(), handleReports() (only the dashboard part).

- **Commit 5:**
    - **Feat(medium):** Implement pricing algorithm and reports
    - **Objective:** Implement Medium-Priority features 5 and 6 (Pricing and Reports).
    - **Classes to program:**
        - model/FinvoryData: Add profitPercentage, discountStandard, discountPremium, discountVip attributes and their getters/setters.
        - model/Product: Implement getPrice() (the dynamic calculation method).
        - model/InvoiceLineSim: Add productId (necessary for the supplier report).
        - FinvoryView: askPriceAlgorithmData(), showReportsMenu(), showReport(), showProductList() (update to display the 3 prices).
        - FinvoryController: handleSetPriceAlgorithm(), handleReportsMenu(), handleSalesReport(), handleCustomerReport(), handleSupplierReport().
        - controller/FinvoryController.handleNewSale(): Update to use p.getPrice(...) instead of the base cost.

- **Task 6:**
    - **Feat (low):** Implement payment methods and Accounts Receivable (A/R)
    - **Objective:** Implement Low-Priority feature 7 (Payment Methods).
    - **Classes to program:**

        - model/InvoiceSim: Add paymentDueDate, and modify status (PENDING/COMPLETED).

        - FinvoryView: askPaymentMethod() (change to Post-Dated Check), askPaymentDueDate(), showCustomerMenu() (add Option 5), choosePendingInvoice().
        - controller/FinvoryController.handleNewSale(): Update to save the invoice as PENDING if it's a check.
        - controller/FinvoryController: Add handleRegisterPayment() (Accounts Receivable logic).

- **Task 7:**
    - **Feat:** Implement login roles (Company/Personal) and validations
    - **Objective:** Implement the two-role login system (based on your CompanyAccount and PersonalAccount classes) and add all validations (Format, Uniqueness, Integrity).

    - **Classes to program:**

        - model/CompanyAccount and model/PersonalAccount: Add username, password fields and checkPassword() method.
        - model/FinvoryData: Add companyAccounts and personalAccounts lists; remove adminInfo.
        - view/FinvoryView: showStartMenu(), showRegistrationMenu(), showLogin(), showPersonalAccountMenu(), showLimitedProductList(), showCompanyPhones(), askNewCompanyAccountData(), askNewPersonalAccountData(). Implement getValidatedStringInput (with Regex) and getPositiveIntInput and update all forms (ask...) to use them.
        - controller/FinvoryController: run(), handleLogin(), handleRegistrationMenu(), handleRegisterCompany(), handleRegisterPersonal(), startPersonalAccountMenu().
        - controller/FinvoryController: Add Uniqueness validations (e.g., findProductByBarcode) in the handleCreate... methods.
        - controller/FinvoryController: Add Integrity validations in handleDeleteCustomer, handleDeleteSupplier, handleDeleteProduct.