**Exercise 4 – Inventory Management**

**Introduction**

The Inventory Management System (IMS) is designed to help manage an inventory consisting of various items, such as products, services, and digital assets. Users can interact with the system through a console-based menu to perform operations like adding, removing, and managing items in the inventory. Additionally, users can calculate prices for items, display their details, and perform stock management functions for products.

**System Design**

The system is structured around a series of classes and methods that ensure modularity and maintainability. Key components include:

1. **InventoryItem (Abstract Class)**
   * This class serves as the base class for all items in the inventory. It contains common properties such as itemId, itemName, and basePrice. It also defines getter methods for these properties.
   * **Methods:**
     + getItemId()
     + getItemName()
     + getBasePrice()
2. **Product (Subclass of InventoryItem)**
   * Represents a physical product with an additional stockLevel attribute.
   * **Methods:**
     + calculatePrice() – Calculates the price of the product based on stock level.
     + displayDetails() – Displays detailed information about the product.
     + getStockLevel() – Retrieves the current stock level of the product.
     + restock(int amount) – Restocks the product by the specified amount.
3. **Service (Subclass of InventoryItem)**
   * Represents a service with an additional serviceDuration attribute.
   * **Methods:**
     + calculatePrice() – Calculates the price of the service based on its duration.
     + displayServiceInformation() – Displays detailed information about the service.
4. **Digital\_Asset (Subclass of InventoryItem)**
   * Represents a digital asset with an additional fileSize attribute.
   * **Methods:**
     + calculatePrice() – Calculates the price of the digital asset based on its file size.
     + displayDigitalAssetInformation() – Displays detailed information about the digital asset.
5. **App (Main Program)**
   * The App class is the main entry point of the system. It provides a console-based menu to the user for interaction with the system.
   * **Features:**
     + Add items to inventory (Products, Services, Digital Assets).
     + Remove items from inventory by specifying the itemId.
     + Display item details based on itemId.
     + Calculate prices for products, services, and digital assets.
     + Perform stock management for products (get stock levels and restock items).
   * The system provides error handling and input validation to ensure smooth user interaction.

**Menu and Functionality**

The system provides the following menu options for user interaction:

1. **Add to Inventory**
   * Allows the user to add a product, service, or digital asset to the inventory by providing relevant details such as itemId, itemName, basePrice, and specific attributes like stockLevel, serviceDuration, or fileSize.
2. **Remove from Inventory**
   * Allows the user to remove an item from the inventory by entering its itemId. If the item is found in the inventory, it is removed; otherwise, an error message is displayed.
3. **Special Functions**
   * **Calculate Price:** The user can enter the itemId and select the item type (product, digital asset, or service) to calculate the price based on item-specific attributes (e.g., stock level, file size, or service duration).
   * **Display Information:** Displays detailed information about a product, service, or digital asset by entering its itemId.
   * **Product Stock Management:** The user can view the stock level of a product and restock it by providing the itemId and the desired restock amount.
4. **Exit Menu**
   * Allows the user to exit the menu and terminate the program.

**Implementation Details**

* **Input Validation:** Throughout the program, input validation ensures that users provide correct and meaningful data. For example, negative prices or invalid stock levels are rejected, and users are prompted to enter valid values.
* **Inventory Management:** Items are stored in an ArrayList<InventoryItem>. This allows for dynamic management of inventory items and supports various operations like adding, removing, and updating items.
* **Polymorphism:** The system leverages polymorphism, especially when handling different item types. The menu option for calculating prices or displaying information checks the instance type (using instanceof) to determine which subclass methods to invoke (i.e., Product, Service, or Digital\_Asset).
* **Restocking and Stock Level Management:** The Product class has functionality to view and update stock levels. Users can check the stock level of any product and restock it as necessary.