**Exercise 3 - Invoice Management**

**Overview:**

The project consists of a system for managing different types of commercial documents, such as invoices, credit notes, and proformas. It involves creating and manipulating these documents while allowing users to add or remove items from an invoice. The system also calculates the total amounts for these documents, incorporating tax rates and refunds.

**Classes & Key Concepts:**

1. **Commercial\_Document (Abstract Class)**
   * The base class for all commercial documents, defining common attributes like document number, date issued, and customer data.
   * The abstract method calculateTotal() is used to enforce a standard calculation method in all subclasses.
2. **Invoice**
   * A subclass of Commercial\_Document, representing a typical invoice.
   * It contains a list of items, each with a price and quantity, and a tax rate.
   * The getInvoiceAmount() method calculates the total amount for the invoice (before tax), and the calculateTotal() method calculates the total including tax.
3. **Credit\_Note**
   * A subclass of Commercial\_Document, representing a credit note issued for a refund.
   * It overrides the calculateTotal() method to print out the refunded amount, based on an associated invoice number.
4. **Proforma**
   * A subclass of Commercial\_Document, representing a proforma document (a preliminary bill).
   * It calculates a validity date based on the number of days input by the user and prints out the validity date in the calculateTotal() method.
5. **Item**
   * A class representing individual items in invoices and other documents.
   * It includes attributes like item code, description, price per unit, and quantity, with methods to retrieve the total cost for the item (getItemAmount()).
6. **Customer**
   * A class representing customer information such as name, surname, phone number, and address.
   * Used in various commercial documents to link the customer to the respective document.
7. **App (Main Class)**
   * The entry point of the application. It allows the user to interact with the system through a menu to add or remove items from an invoice, and create invoices, credit notes, and proforma documents.
   * The program collects customer details, item details, and generates different types of commercial documents based on user input.

**Challenges & Solutions:**

1. **Dynamic User Input:**
   * Handling user input, especially for item details and document creation, required careful validation to ensure correct data types and avoid errors (e.g., negative or invalid numbers).
   * The use of Scanner for reading inputs is effective for a simple prototype, but might be improved by adding input validation and error handling in future iterations.
2. **Document Calculation:**
   * The challenge of managing multiple types of commercial documents (invoice, credit note, proforma) was tackled by utilizing inheritance and polymorphism. Each document class overrides the calculateTotal() method to provide specific functionality while maintaining a common interface.