Internship - Mobile

Please note the following:

1. You can use any programming language you want.
2. You can use any platform you want (mobile, desktop or web).
3. Note that some exercises can benefit from the interaction with the user of the program.
    a. You can implement any type of UI you want (command line, GUI or any type you are comfortable with).
4. You can use any persistency mechanism you want (files, database, system preferences etc.)
5. Please provide minimal instructions on how to run the program.
6. If you created anything to visualise the design of a solution, please share it with us.
7. You can share the solution using any channel you want (public repository, email archive, transfer link).
8. We encourage you to send your solution even if it solves a partial number of subtasks.

Invoice Manager

Write a small program which provides functionalities for administrating invoices.

Mainly we want to be able to generate, order, filter and edit a list of invoices.

The following **Business Objects** are involved:

Company, representing a commercial entity. It has the following properties:

- name
- phone number

Product, representing a commercial product. It has the following properties:

- product number (unique identifier)
- name
- price

Invoice, representing a fiscal proof of a commercial intend. It has the following properties:

- invoice number (unique identifier)
- seller
- products
- total
- due date
- pay date (optional)

Subtasks

Subtask 1: Generate Companies

1. Define a set of 8 strings that will be used later to represent first names of the companies.
    a. *E.g. "Romanian", "European", "Food", "Electricity", "Incorporated" ...*
2. Generate a set of 24 ***Companies*** with the following rules:
    a. company name should be composed from 2 or 3 strings from the pool of names (use space as separator).
    b. at least 3 companies should have 3 strings in their name: E.g. "Romanian Food Incorporated"
    c. all strings appearing in a company name should be different.
    d. no full company name should be repeated.
    e. the phone number can be a random number of 14 digits.

Subtask 2: Generate Products

1. Generate a set of 48 ***Products*** with the following rules:
    a. product number is unique.
    b. name is composed of 5 random alphabetical characters (please use the English alphabet).
    c. each name should be unique.
    d. price is a random number between 0.1 and 999.9.

Subtask 3: Generate Invoices

1. Generate a set of 50 **Invoices** with the following rules:
    a. all companies generated at **Subtask no. 1** must be present as seller in at least one invoice.
    b. each invoice must contain at least one product generated in **Subtask no. 2** but no more than 3 products.
    c. total represents the sum of prices for the products involved in the invoice.
    d. due date is any date in the next 5 days. **Time of day is optional.**
    e. pay date is any date in the previous 5 days (including the current day). **Time of day is optional.**
        i. pay date is optional. If pay date is missing, the invoice is not paid yet ("unpaid").
        ii. future dates are not allowed. If you choose to ignore time, current day should not be considered as future date.
    f. at least 10% of the generated invoices must be "unpaid".

g. invoice number must be unique.

Subtask 4: Equality vs. Identity

1. If two invoices have equal properties, except for invoice number, these are considered ***equal***.
2. Add an extra invoice which is *equal* with an existing one. You can do this directly via code after code for Subtask 3 is completed. UI is optional.
3. Accept this new invoice but mark it as a "duplicate".

Subtask 5: Ordered invoices

1. Output the invoices in an orderly manner:
    a. Pending invoices should appear first, ordered by due date (closest date first).
    b. Show how many days are left until the payment is due.
    c. Pending invoices are followed by the paid invoices, ordered by paid date (latest date first).
        i. Show date when invoice was paid.
        ii. If any duplicated invoice are present, add a marker indicating that an invoice is a duplicate.
2. Properties needed in the output:
    a. invoice number
    b. seller
    c. dates
    d. products (names only)
    e. total.
    f. duplication flag (if the invoice is a duplicated one).

Subtask 6: Text Search

1. Filter the invoice list based on a give text (quick text search):
    a. Text must be received from the user of the program.
    b. Text should contain at least 3 characters.
2. List of results should be order as such:
    a. Invoices with the first word in seller name matching with the given text.
    b. Invoices with the second word in seller name matching with the given text.
    c. Invoices with any content of seller name matching with the given text.
    d. Invoices with any product name matching with the given text.
3. List of results should contain 10 items or less.
4. Properties needed in the output:
    a. seller name
    b. dates
    c. product (names only).
    d. try to highlight the text match. This depends on the type of UI you want to use. **(Optional)**

Subtask 7: Editing

1. Allow the user of the program to "pay" a specific invoice.
2. Depending on the UI you write, let the user specify an invoice he/she wants to mark as paid.
3. This will update the "pay date" property of the invoice with the current day. Remember that pay date is an optional property.

Subtask 8: Persistency

1. Persist the generated data between program executions.
2. Data to be persisted: Companies, Product, Invoices.
3. At the beginning of the program, let the user choose if he/she wants to use the persisted data or generate a new one.

Subtask 9: Complete User Flow

Note that you can complete the previous subtasks without writing any UI. Any alternative channels for input and output are accepted.

For the final task, please provide a UI for the input and output of the features. This will replace the alternative channels used. As mentioned at the beginning of the paper, you can choose any type of UI you want but try to fulfil the requirements from below.

Implement the complete user flow as following:

1. At the beginning of the program, user must first choose to with a new set of data or use the persisted data (taken from Subtask no. 8).
    a. show this option only if persisted data is available.
    b. if the user chooses to generate new data please jump to **step 2.**
    c. if the user chooses to use persisted data, please jump to **step 6**.
2. Let the user choose to input the set of Company names or use the predefined ones (from Subtask no.).
3. Let the user input the number of Companies that will be generated (used at Subtask no. 1).
4. Let the user input the number of Products that will be generated (used at Subtask no. 2).
5. Let the user input the number of Invoices that will be generated (used at Subtask no. 3).
6. Let the user choose one of the following:
    a. to see the ordered invoice list (from Subtask no. 5).
    b. to duplicate an invoice (from Subtask no. 4).
    c. search the list by giving a text (from Subtask no. 6).
    d. mark an invoice as paid (from Subtask no. 7).
    e. persist the current data (from Subtask no. 8).
7. After each operation the user should be able to repeat the choice at step 7.