BNPL

Problem Definition:

Buy Now Pay Later (BNPL) is a product where a customer can buy the product instantly and pay the dues within **30 days** as per his convenience. Each user has a credit limit associated with them initially.

Can you help us implement the concept?

Write an application that does the following:

1. **seed_inventory**: The application should read from a FILE/STDIN/DRIVER and add products to inventory with relevant attributes (like name, count, price etc.) and store it in memory.
2. **view_inventory**: The application should give a view of inventory at any given point of time. One should be able to sort the view based on its attributes in ascending order.
3. **register_user** : Initialise an user with basic details and initial BNPL credit limit.
4. **buy**: As a customer you should be able to buy the products available in the inventory. The order gets placed, the BNPL credit limit decreases, the order gets added to customer order history and inventory gets reduced. If the credit limit is exhausted, payment should not be allowed.
   API signature : buy(user, List<item, quantity>, payment_method, date_of_purchase)
   Payment can be of 2 types:
   a. **PREPAID**: The order gets placed and inventory gets reduced.
   b. **BNPL**: The order gets placed, the BNPL credit limit decreases, and inventory gets reduced. If the credit limit is exhausted, payment should not be allowed.
5. **clear_dues**: A user at any time can choose to pay the dues (partial or complete) pending on him. Partial payment refers to clearing of dues of a few of the orders. Users can choose few (or all) the orders they wish to pay for at this time.
   API signature : clear_dues(user, List<order>, date_of_clearing)

Bonus [Optional]

(Attempt only after completing above functionalities)

1. **view_dues :** A user can view all the dues pending at his name along with the order which the due belongs to. The API should print the date of purchase, pending amount, order detail. Print all the dues in ascending order of date of purchase. Also print the status of due -> DELAYED (if 30 day window is crossed) OR PENDING.
   API signature : view_dues(user, date)

2. **blacklisting:** If a customer defaults payment for 3 orders, he should be blacklisted and he won't be able to place new orders using BNPL.

Note: Expectations:

1. We expect you to store all data in memory. Usage of the database to store the contents is also not allowed.
2. Create the sample data yourself. You can put it into a file, test case or main driver program itself or use STDIN. You should be able to modify the test cases when asked.
3. Code should be **demo-able.** Either by using a main driver program or test cases.
4. Code should be modular. Code should have basic OO design.
5. Code should be extensible. Wherever applicable, use interfaces and contracts between different methods. It should be easy to add/remove functionality without re-writing the entire codebase.
6. Code should handle edge cases properly and fail gracefully.
7. Code should be legible, readable and DRY.
8. You can choose to show the final output on STDOUT or write to a file.

Guidelines:

- Time: 90mins
- Write modular and clean code.
- **A driver program/main class/test case is needed to test out the code by the evaluator with multiple test cases.** But do not spend too much time in the input parsing. Keep it as simple as possible.
- Evaluation criteria: Demoable & functionally correct code, Code readability, Proper Entity modeling, Modularity & Extensibility, Separation of concerns, Abstractions. Use design patterns wherever applicable

- You are not allowed to use any external databases like MySQL. Use only in memory data structures.
- No need to create any UI
- Please focus on the Bonus Feature only after ensuring the required features are complete and demoable.

Sample Examples

Only basic details are listed in example as input. You can take in more details while designing. For example: a user can have more entities like mobile number etc.

```
-> seed_inventory(“Shoes 5 200” , “Watch 10 1000”, “T-Shirt 14 300”)

-> view_inventory(“name”)
     1. Shoes 5 200
     2. T-Shirt 14 2000
      3. Watch 10 1000

-> view_inventory(“price”)
     1. Shoes 5 200
     2. Watch 10 1000
     3. T-Shirt 14 2000

-> register_user(“Akshay”, 5000)

-> buy(“Akshay”, (<Shoes,2>,<Watch,1>), “BNPL”, “20-Oct-2021”)

-> view_inventory()
     1. Shoes 3 200
     2. Watch 9 1000
     3. T-Shirt 14 2000

-> view_dues(“Akshay”, “21-Nov-2021”)
   1. 20-Oct-2021
     1400
     Due By : <20-Nov-2021>
     DELAYED
     (Also list other important information as per your
     understanding)

-> clear_dues(“Akshay”, List_of_order, “19-Nov-2021”)

-> view_dues(“Akshay”, “20-Nov-2021”)
 [None]
```