Flipkart BNPL

-> order_status(0001)
3600(BNPL Credit limit available)
20-Oct-2021 BNPL <Shoes,2>,<Watch,1> 1400(amount)

Problem Definition:

Flipkart is known for its innovative products and making shopping easier for customers. One such innovation is Buy Now Pay Later (BNPL), 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.
   API signature : view_inventory()

3. **register_user** : Initialize an user with basic details and initial BNPL credit limit.
   API signature : register_user(user, credit_limit)

4. **buy**: As a customer you should be able to buy the products available in the inventory.
   API signature : buy(user, List<item, quantity>, payment_method, date_of_purchase)
   Payment can be of 2 types:
   a. **PREPAID**: The order gets placed & added to order history and inventory gets reduced.
   b. **BNPL**: 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.

Note: *buy* method can return an orderId which can be used in subsequent functionalities.
c

5. **clear_dues**: An 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<orderId>, date_of_clearing)

6. **view_dues** : A user can view all the dues pending at his name (for the orders placed before the given date) 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)

7. **order_status**: At any point, order history should be available to view.(Can be printed in ascending order of date of purchase) Also print the current BNPL credit limit available.
   API signature : order_status(user)

Bonus

(Attempt only after completing above functionalities)

1. blacklisting [BONUS]: If a customer defaults payment for 3 orders, he should be blacklisted and he shouldn't be able to place new orders using BNPL.(Add the blacklisting check in the buy method, so that it checks whether the user should be blacklisted before allowing the order to be placed)

2. add_inventory/remove_inventory[BONUS]: The application should allow you to add/remove inventory items.

Note: We expect you to store all data in memory. Usage of the database to store the contents is also not allowed.

Expectations:

1. 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.

2. Code should be **demo-able**. Either by using a main driver program or test cases.
3. Code should be modular. Code should have basic OO design.
4. 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.
5. Code should handle edge cases properly and fail gracefully.
6. Code should be legible, readable and DRY.
7. You can choose to show the final output on STDOUT or write to a file.

Guidelines:

1. Make sure you get any clarification you need before starting the implementation.
2. Please discuss the solution with an interviewer.
3. Please do not access the internet for anything EXCEPT syntax.
4. You are free to use the language of your choice,
5. All work should be your own.
6. Overall time: 90 mins

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 2000")
W14
-> view_inventory()
     1. Shoes 5 200
     2. Watch 10 1000
     3. T-Shirt 14 2000

-> register_user("Akshay", 5000)
-> Assuming above registered user has user_id = 0001

-> buy(0001, (<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(0001, “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(0001, <”OD_123”> , “19-Nov-2021”)

-> view_dues(0001, “20-Nov-2021”)
 [None]
```