Assignment 3: Byte Me!

Assumptions:

1. The menu is initially empty. Only admin can add items to the menu.
2. When the customer chooses to checkout, they are given a prompt which asks them if they would like to pay to become a VIP customer. If they say "yes", then their order is made a priority and is handled before the orders of the regular customers (but their order ID (which denotes their number in the queue without the consideration of their status) remains the same.
3. Every time a customer checks out, they are asked if they'd like to pay to become a VIP customer.
4. An order is placed by the customer only after their checkout is successful.
5. Once the checkout is successful, their cart is made empty.
6. An order can have one of the following as their status:
    1. Order Received
    2. Order Being Prepared
    3. Order Out for Delivery
    4. Order Delivered and Payment Received
    5. Refund Initiated
    6. Refunded
7. The status of an order is set by default to "Order Received".
8. When the status of an order is set to "Order Delivered and Payment Received", it still shows in pending orders.
9. When a customer chooses to cancel their order, the status is set to "Refund Initiated", and that order is not prepared, that is, it does not show in pending orders.
10. An order is considered an 'order of the day' if its status is "Order Delivered and Payment Received".
11. When the admin chooses to handle special requests, they can see the orders with special requests. No action can be done on them.

Use of Java Collections

1. HashMap<String, User> : Used in App.java to store users with usernames as keys and User objects as values.
2. ArrayList<Item> menuItems : Holds all menu items available in an arraylist.
3. List<Order> orderQueue : List of all regular orders to be processed.
4. List<Order> vipOrderQueue : List for VIP orders to be processed.
5. List<Order> regularCancelled> : List for regular cancelled orders.
6. List<Order> vipCancelled : List for VIP cancelled orders.
7. Map<String, Integer> cart : Used to store the items in a customer's cart, along with the quantity of those items.
8. Map<Integer, Order> orderHistory : Stores the order history, where the ID is the key and Order object is the value.
9. Map<String, String> specialRequests : Holds special requests as a string for the items in the cart.
10. Map<String, Integer> items : Stores the items within an order, with item names as keys and their quantities as values.
11. Map<String, String> specialRequests : Keeps track of special requests of an items in an order.

Used OOP principles:

1. Encapsulation
2. Inheritance
3. Polymorphism
4. Abstraction
5. Composition