**Milestone 1: Item Class Hierarchy**

**Objective:** Create a class hierarchy to represent different item types.

**Tasks:**

1. **Create Base Class Item:**
   * Define common attributes:
     + String id (unique identifier).
     + String name (name of the item).
     + double price (price of the item).
     + int quantity (quantity in stock).
   * Implement constructors and getter/setter methods.
2. **Create Subclasses for Specific Item Types:**
   * **Electronics**: Add int warranty attribute.
   * **Clothing**: Add String size attribute.
   * **Book**: Add String author attribute.
3. **Implement Comparable<Item> Interface:**
   * Learning Resource : [Comparable Interface in Java with Examples- Scaler Topics](https://www.scaler.com/topics/comparable-interface-in-java/)
   * Write a compareTo method to sort items alphabetically by name.
4. **Write a Test Program for Subclasses:**
   * Instantiate a few items for each subclass.
   * Add them to a list and test sorting by name.

**Milestone 2: Generic Inventory Management**

**Objective:** Implement a generic inventory system for managing items.

**Tasks:i**

1. **Design the Inventory<T extends Item> Class:**
   * Use HashMap<String, T> to store items, where id is the key.
2. **Add Core Methods to Inventory:**
   * addItem(T item): Add an item to the inventory.
   * removeItem(String id): Remove an item by id.
   * getItem(String id): Retrieve an item by id.
   * getAllItems(): Return all items as a Collection<T>.
3. **Write Tests for Inventory:**
   * Add, retrieve, and remove items.
   * Verify behavior when trying to add items with duplicate IDs.

**Milestone 3: Recently Viewed Items**

**Objective:** Maintain a list of recently viewed items using LinkedList.

**Tasks:**

1. **Use LinkedList<Item> for Recently Viewed Items:**
   * Create a LinkedList<Item> to store recently viewed items.
   * You can also use LinkedHashMap : Figure it out!
2. **Add Helper Methods:**
   * addRecentlyViewedItem(Item item):
     + Add the item to the list. Explore [methods](https://www.scaler.com/topics/linked-list-in-java/#:~:text=Methods%20for%20Linked%20List%20in%20Java) available in LinkedList to implement this.
     + Ensure the list size doesn’t exceed 10. Remove the oldest item if exceeded.
3. **Test Recently Viewed List:**
   * Add more than 10 items and ensure the oldest items are removed correctly.

**Milestone 4: Order Processing with Priority Queue**

**Objective:** Process orders by prioritizing express orders using a PriorityQueue.

**Tasks:**

1. **Create the Order Class:**
   * Attributes:
     + String orderId.
     + boolean isExpress (true for express orders, false for regular ones).
2. **Implement the Order Queue:**
   * Use PriorityQueue<Order> to store orders.
3. **Write Methods for Order Queue:**
   * addOrder(Order order): Add an order to the queue.
   * processOrder(): Process and remove the highest-priority order (express orders first).
4. **Test Order Queue:**
   * Add multiple express and regular orders.
   * Test if express orders are processed first.

**Milestone 5: Sorting and Filtering Options**

**Objective:** Enable sorting and filtering of items based on different criteria.

**Tasks:**

1. **Implement Custom Comparators:**
   * Write Comparators for sorting by:
     + price.
     + quantity.
2. **Add Filtering Methods in Inventory Class:**
   * filterByPriceRange(double minPrice, double maxPrice): Return items within a price range.
   * filterByAvailability(): Return items with quantity > 0.
3. **Test Sorting and Filtering:**
   * Test sorting and filtering methods with a list of items.

**Milestone 6: Customer Wishlist**

**Objective:** Manage a customer’s wishlist using a Set to ensure uniqueness.

**Tasks:**

1. **Use Set<Item> for Wishlist:**
   * Create a Set<Item> to store the wishlist.
2. **Add Wishlist Methods:**
   * addToWishlist(Item item): Add an item to the wishlist.
   * removeFromWishlist(Item item): Remove an item from the wishlist.
3. **Test Wishlist Functionality:**
   * Add duplicate items and verify that only unique items are stored.