**Unit Testing Description**

**EcomSystemTest Class:**

This test class is for testing an e-commerce system.

1. **Test Login**: First, we're testing if users can sign up and login. We're creating a user named Ahmed, and we're checking if we can log in with the username "Dino" and password "Danielo". We also check if someone with the wrong username and password can't log in.
2. **Test Signup**: Next, we're testing the signup process. We create a user named Siveen and check if it's successful. Then, we try to create the same user again to make sure it fails, as it should. Finally, we check if the user we created has the correct username and password.
3. **Test Get Current User**: Lastly, we're testing if we can get the current user. At the beginning, there shouldn't be any current user, so we check if it's null. Then, we sign up a user named Chery and make sure we can get the current user, and that it's the same as the one we signed up.

These tests help ensure that the e-commerce system is working as expected, allowing users to sign up, log in, and access their accounts correctly.

**UserTest Class:**

This test class is for testing the functionality of the user class in an e-commerce system.

1. **Order Management Tests**:
   * **Test Add Order**: This test checks if an order can be added to the user's order list. It creates an order, adds it to the user, and then checks if the user's order list is not null and has a size of 1.
   * **Test Order Exists**: This test verifies if the user can retrieve an order by its ID after adding it. It creates an order, adds it to the user, and then checks if the retrieved order matches the added one.
   * **Test Getters**: This test validates various getters of the user object, including checking if the username, cart, and password are set correctly.
2. **Visa User Management Tests**:
   * **Test Add Visa**: This test ensures that the user can add multiple visas and that the list of added visas matches the expected list. It creates three visa objects, adds them to the user, and compares the expected and actual lists.
   * **Test Username**: This test simply checks if the username retrieved from the user object matches the one set during setup.

.

**OrderTest Class:**

This test class is designed to validate the functionality of the order class in an e-commerce system

1. **Order Creation Tests**:
   * **Test Order ID Generation**: Checks if the order ID is generated properly. It creates two orders and ensures that their IDs are not null and follow the expected format.
   * **Test Total Price Calculation**: Verifies if the total price of the order is calculated correctly. Initially, it checks if the total price matches the expected value. Then, it adds another item to the cart and verifies that the total price changes accordingly.
   * **Test Order Status**: Ensures that the initial status of the order is "Order\_Placed".
   * **Test Time Difference in Seconds**: Validates the method that calculates the time difference in seconds. It schedules an end time 30 seconds from the current time and checks if the difference is correctly calculated.
   * **Test Schedule Status Update**: Tests if the status of the order changes to "Dispatch" after scheduling a status update.
2. **Stock Handling**:
   * **Test Cancel Order**: Checks if canceling an order restores the stock of items in the cart to their initial quantities.
   * **Test Remove from Stock**: Verifies if removing items from stock during order placement updates the stock quantities correctly.
3. **Test Getters and Setters**: Ensures that the getters and setters of the **Order** class work as expected. It validates if the cart, order ID, address, and phone number can be retrieved correctly, and if the status can be updated successfully.

**CartTest Class**:

This CartTest class is focused on testing the functionalities of a shopping cart in an e-commerce system.

1. **Add To Cart Tests**:
   * **Valid and Invalid Amount**: these tests verify whether items can be added to the cart with valid and invalid quantities. The first test checks if items can be added successfully within the available stock limit, while the second test ensures that attempting to add more items than available stock results in an empty cart.
2. **Remove From Cart Test**:
   * This test checks if items can be successfully removed from the cart. It adds an item, removes it, and then verifies that the cart is empty and the total price is zero.
3. **Amount Increase and Decrease Tests**:
   * These tests validate the functionality to increase and decrease the quantity of items in the cart. They add items, increase or decrease their quantities, and verify if the changes are reflected correctly in the cart's contents and total price.
4. **Empty Cart Test**:
   * This test ensures that the cart can be emptied successfully. It adds items to the cart, empties it, and then verifies that the cart is indeed empty and the total price is zero.
5. **Tests for Getters**:
   * these tests validate the getters of the cart class, They add items to the cart, retrieve its contents and total price, and assert that they match the expected values.

**VisaTest Class:**

This visa Test class is responsible for testing the functionality of the visa class, particularly focusing on validation checks for Visa card numbers and CVV (Card Verification Value).

1. **Test Valid Visa**:
   * This test ensures that the method correctly validates the expiration date of the Visa card. It checks if dates within the valid range return true and dates outside the valid range return false.
2. **Test Invalid Visa - Year, Month**:
   * This parameterized test verifies that method correctly handles invalid combinations of year and month. It uses a CSV source to provide different combinations of year and month, testing if the method returns false for all invalid cases.
3. **Visa CVV Check**:
   * This test validates the method. It checks if the method returns true for a correct CVV and false for an incorrect one.
4. **Visa Number Check**:
   * This test verifies that the Visa card number stored in the Visa object matches the expected value.

**CatalogTest Class:**

This class is designed to test the functionality of the Catalog class.

1. **Add Item Test**:
   * This test validates the method. It adds two items to the catalog and checks if the size of the catalog increases accordingly and if the added items are present in the catalog.
2. **Tests for Getters**:
   * + **Get Item Test**: Verifies if the method retrieves the correct item from the catalog based on its name.
     + **Get All Items Test**: Checks if the method returns a list containing all items in the catalog and if it includes the items that were added.
3. **Size Test**:
   * This test ensures that the method correctly returns the number of items in the catalog. It verifies if the size is 0 initially and if it increases after adding items.

**ItemTest Class:**

This class is responsible for testing the functionality of the Item class, which likely represents individual items available for sale in an e-commerce system. Let's review the tests:

1. **Tests for Order and Unorder Methods**:
   * + **Test for Order Method with Valid Amount**: Verifies if the ordered() method correctly decreases the stock of the item by the specified amount.
     + **Test for Unorder Method**: Checks if the unordered() method correctly increases the stock of the item by the specified amount.
2. **Tests for Getters**:
   * + **Get Name Test**: Ensures that the getName() method returns the correct name of the item.
     + **Get Price Test**: Verifies that the getPrice() method returns the correct price of the item.
     + **Get Stock Test**: Checks if the getStock() method returns the correct stock quantity of the item.
     + **Get Image Test**: Validates that the getImage() method returns the correct image path of the item.