Project 4

Sure, let's continue by modifying the `ProductTester` class to include static methods for processing, and then implementing the user interface with a menu system.

Step 1: Create Static Methods in `ProductTester` Class

1. **Create `displayInventory` method**:

  - This method will display the inventory of products.

2. **Create `addToInventory` method**:

  - This method will populate the product array with user input.

3. **Create `getNumProducts` method**:

  - This method will get the number of products from the user.

4. **Move relevant code into these methods**:

  - Refactor the main method to call these new methods.

Modified `ProductTester` Class

```java
import java.util.Scanner;

public class ProductTester {
    public static void main(String[] args) {
        // Create a Scanner object for keyboard input
        Scanner in = new Scanner(System.in);

        int maxSize = getNumProducts(in);
    if (maxSize == 0) {
        System.out.println("No products required!");
    } else {
        Product[] products = new Product[maxSize];
        addToInventory(products, in);

        int menuChoice;
        do {
            menuChoice = getMenuOption(in);
            executeMenuChoice(menuChoice, products, in);
        } while (menuChoice != 0);
    }

    // Close the Scanner
    in.close();
}

// Method to display the inventory
public static void displayInventory(Product[] products) {
    System.out.println("\nProduct details:");
    for (Product product : products) {
        System.out.println(product.toString());
    }
}

// Method to add products to the inventory
public static void addToInventory(Product[] products, Scanner in) {
    int tempNumber;
    String tempName;
    int tempQty;
    double tempPrice;
```

```
        for (int i = 0; i < products.length; i++) {
            in.nextLine(); // Clear the input buffer
            System.out.println("Enter the details for Product " + (i + 1) + ":");
            System.out.print("Item Number: ");
            tempNumber = in.nextInt();
            in.nextLine(); // Consume newline left-over
            System.out.print("Name: ");
            tempName = in.nextLine();
            System.out.print("Quantity: ");
            tempQty = in.nextInt();
            System.out.print("Price: ");
            tempPrice = in.nextDouble();

            // Create a new product and add it to the array
            products[i] = new Product(tempNumber, tempName, tempQty, tempPrice);
        }
    }

    // Method to get the number of products from the user
    public static int getNumProducts(Scanner in) {
        int maxSize = -1; // Initial value to fail the loop initially

        // Prompt user for number of products
        do {
            try {
                System.out.print("Enter the number of products you would like to add (Enter 0 if you do not
wish to add products): ");
                maxSize = in.nextInt();
                if (maxSize < 0) {
                    System.out.println("Incorrect Value entered");
            }
        } catch (Exception e) {
            System.out.println("Incorrect data type entered! Please enter a valid integer.");
            in.nextLine(); // Clear the input buffer
        }
    } while (maxSize < 0);

    return maxSize;
}

// Method to get the menu option from the user
public static int getMenuOption(Scanner in) {
    int menuChoice = -1;
    do {
        try {
            System.out.println("\nMenu:");
            System.out.println("1. View Inventory");
            System.out.println("2. Add Stock");
            System.out.println("3. Deduct Stock");
            System.out.println("4. Discontinue Product");
            System.out.println("0. Exit");
            System.out.print("Please enter a menu option: ");
            menuChoice = in.nextInt();
            if (menuChoice < 0 || menuChoice > 4) {
                System.out.println("Invalid menu option. Please enter a number between 0 and 4.");
            }
        } catch (Exception e) {
            System.out.println("Incorrect data type entered! Please enter a valid integer.");
            in.nextLine(); // Clear the input buffer
        }
    } while (menuChoice < 0 || menuChoice > 4);
    return menuChoice;
}

// Method to execute the menu choice
public static void executeMenuChoice(int menuChoice, Product[] products, Scanner in) {
    switch (menuChoice) {
        case 1:
            System.out.println("View Product List");
            displayInventory(products);
            break;
        case 2:
            System.out.println("Add Stock");
            addInventory(products, in);
            break;
        case 3:
            System.out.println("Deduct Stock");
            deductInventory(products, in);
            break;
        case 4:
            System.out.println("Discontinue Stock");
            discontinueInventory(products, in);
            break;
        case 0:
            System.out.println("Exiting...");
            break;
        default:
            System.out.println("Invalid choice!");
            break;
    }
}
```

```
    // Method to get the product number from the user
    public static int getProductNumber(Product[] products, Scanner in) {
        int productChoice = -1;
        do {
            try {
                System.out.println("\nSelect a product:");
                for (int i = 0; i < products.length; i++) {
                    System.out.println(i + ": " + products[i].getName());
                }
                System.out.print("Enter the product number: ");
                productChoice = in.nextInt();
                if (productChoice < 0 || productChoice >= products.length) {
                    System.out.println("Invalid product number. Please enter a number between 0 and " +
(products.length - 1));
                }
            } catch (Exception e) {
                System.out.println("Incorrect data type entered! Please enter a valid integer.");
                in.nextLine(); // Clear the input buffer
            }
        } while (productChoice < 0 || productChoice >= products.length);

        return productChoice;
    }

    // Method to add stock to a product
    public static void addInventory(Product[] products, Scanner in) {
        int productChoice = getProductNumber(products, in);
        int updateValue = -1;

        do {
            try {
                System.out.print("How many products do you want to add? ");
                updateValue = in.nextInt();
                if (updateValue < 0) {
                    System.out.println("Invalid value. Please enter a positive number.");
                }
            } catch (Exception e) {
                System.out.println("Incorrect data type entered! Please enter a valid integer.");
                in.nextLine(); // Clear the input buffer
            }
        } while (updateValue < 0);

        products[productChoice].addToInventory(updateValue);
    }

    // Method to deduct stock from a product
    public static void deductInventory(Product[] products, Scanner in) {
        int productChoice = getProductNumber(products, in);
        int updateValue = -1;

        do {
            try {
                System.out.print("How many products do you want to deduct? ");
                updateValue = in.nextInt();
                if (updateValue < 0 || updateValue > products[productChoice].getQuantity()) {
                    System.out.println("Invalid value. Please enter a number between 0 and " +
products[productChoice].getQuantity());
                }
            } catch (Exception e) {
                System.out.println("Incorrect data type entered! Please enter a valid integer.");
                in.nextLine(); // Clear the input buffer
            }
        } while (updateValue < 0 || updateValue > products[productChoice].getQuantity());

        products[productChoice].deductFromInventory(updateValue);
    }

    // Method to discontinue a product
    public static void discontinueInventory(Product[] products, Scanner in) {
        int productChoice = getProductNumber(products, in);
        products[productChoice].setActive(false);
    }
}
```

Step 2: Modify `Product` Class to Include New Methods

1. **Create `addToInventory` method**:
   - This method will add stock to the inventory.

2. **Create `deductFromInventory` method**:
   - This method will deduct stock from the inventory.

Modified `Product` Class

```java
public class Product {
    private int itemNumber;
    private String name;
    private int quantity;
    private double price;
    private boolean active;
```

```
public Product(int itemNumber, String name, int quantity, double price) {
   this.itemNumber = itemNumber;
   this.name = name;
   this.quantity = quantity;
   this.price = price;
   this.active = true;
}

public int getItemNumber() {
   return itemNumber;
}

public String getName() {
   return name;
}

public int getQuantity() {
   return quantity;
}

public double getPrice() {
   return price;
}

public boolean isActive() {
   return active;
}

public void setActive(boolean active) {
   this.active = active;
```

```
    }

    public void addToInventory(int quantity) {
        this.quantity += quantity;
    }

    public void deductFromInventory(int quantity) {
        this.quantity -= quantity;
    }

    @Override
    public String toString() {
        return "Item Number: " + itemNumber +
                "\nName: " + name +
                "\nQuantity in stock: " + quantity +
                "\nPrice: " + price +
                "\nStock Value: " + getInventoryValue() +
                "\

nActive: " + active;
    }

    public double getInventoryValue() {
        return quantity * price;
    }
}
```

Step 3: Run and Test Your Code

Run your program to ensure it works as expected. This completes the necessary updates for Section 7 Part 1 of the inventory project. If you have any issues or need further assistance, feel free to ask!