Scenario

You are working for a small manufacturing organization that has outgrown its current inventory system. Members of the organization have been using a spreadsheet program to manually enter inventory additions, deletions, and other data from a paper-based system but would now like you to develop a more sophisticated inventory program.

You have been provided with a mock-up of the user interface to use in the design and development of the system (see the attached “Software 1 GUI Mock-Up”) and a class diagram to assist you in your work (see the attached “UML Class Diagram”). The organization also has specific business requirements that must be considered for the application. A systems analyst created the solution statements outlined in the requirements section of this task based on the business requirements. You will use these solution statements to develop your application.

Requirements

I. User Interface

A. Create a JavaFX application with a graphical user interface (GUI) based on the attached “Software 1 GUI Mock-Up.” You may use JavaFX with or without FXML to create your GUI, or you may use Scene Builder to create your FXML file; use of Swing is not permitted. The user interface (UI) should closely match the organization of the GUI layout and contain *all* UI components (buttons, text fields, etc.) in *each* of the following GUI mock-up forms:

1. Main form
2. Add Part form
3. Modify Part form
4. Add Product form
5. Modify Product form

Note: You may use one FXML file for forms with an identical UI component structure. You may also use a single window that can be switched to a different menu, or a new window can be launched for each form. As of JDK 11, JavaFX is no longer included in the JDK API but is available as an SDK or module.

B. Provide Javadoc comments for *each* class member throughout the code, and include a detailed description of the following in your comments:

• a logical or runtime error that you corrected in the code and how it was corrected

• a future enhancement that would extend the functionality of the application if it were to be updated

Note: For these comments to accurately export to the Javadoc comments, please add the logical and runtime error comments in the method header declaration comments where the error that was corrected occurred, and include the future enhancement comments in the comments of the main class. Please start these comments with “RUNTIME ERROR” or “FUTURE ENHANCEMENT” as applicable.

II. Application

C. Create classes with data and logic that map to the UML class diagram and include the supplied Part class provided in the attached "Part.java." Do not alter the provided class. Include all the classes and members as shown in the UML diagram. Your code should demonstrate the following:

- inheritance
- abstract and concrete classes
- instance and static variables
- instance and static methods

D. Add the following functionalities to the Main form:

1. The Parts pane
   - The Add button under the Parts TableView opens the Add Part form.
   - The Modify button under the Parts TableView opens the Modify Part form.
   - The Delete button under the Parts TableView deletes the selected part from the Parts TableView or displays a descriptive error message in the UI or in a dialog box if a part is not deleted.
   - When the user searches for parts by ID or name (partial or full name) using the text field, the application displays matching results in the Parts TableView. (Including a search button is optional.) If the part or parts are found, the application highlights a single part or filters multiple parts. If the part is not found, the application displays an error message in the UI or in a dialog box.
   - If the search field is set to empty, the table should be repopulated with all available parts.
2. The Products pane
   - The Add button under the Products TableView opens the Add Product form.
   - The Modify button under the Products TableView opens the Modify Product form.
   - The Delete button under the Products TableView deletes the selected product (if appropriate) from the Products TableView or displays a descriptive error message in the UI or in a dialog box if a product is not deleted.
   - When the user searches for products by ID or name (partial or full name) using the text field, the application displays matching results in the Products TableView. (Including a search button is optional.) If a product or products are found, the application highlights a single product or products or filters multiple products. If a product or products are not found, the application displays an error message in the UI or in a dialog box.
   - If the search field is set to empty, the table should be repopulated with all available products.

   Note: A product's associated parts can exist independent of current inventory of parts. You are not required to display sample data upon launching your application. You do not need to save your data to a database or a file; data for this application is nonpersistent and will reside in computer memory while in use.

3. Exit button

• The Exit button closes the application.

E. Add the listed functionalities to the following parts forms:

1. The Add Part form

• The In-House and Outsourced radio buttons switch the bottom label to the correct value (Machine ID or Company Name).

• The application auto-generates a unique part ID. The part IDs can be, but do not need to be, contiguous.

- The part ID text field must be disabled.

• The user should be able to enter a part name, inventory level or stock, a price, maximum and minimum values, and company name or machine ID values into active text fields.

• After saving the data, users are automatically redirected to the Main form.

• Canceling or exiting this form redirects users to the Main form.

2. The Modify Part form

• The text fields populate with the data from the chosen part.

• The In-House and Outsourced radio buttons switch the bottom label to the correct value (Machine ID or Company Name) and swap In-House parts and Outsourced parts. When new objects need to be created after the Save button is clicked, the part ID should be retained.

• The user can modify data values in the text fields sent from the Main form except the part ID.

• After saving modifications to the part, the user is automatically redirected to the Main form.

• Canceling or exiting this form redirects users to the Main form.

F. Add the following functionalities to the following product forms:

1. The Add Product form

• The application auto-generates a unique product ID. The product IDs can be, but do not need to be, contiguous.

- The product ID text field must be disabled and cannot be edited or changed.

• The user should be able to enter a product name, inventory level or stock, a price, and maximum and minimum values.

• The user can search for parts (top table) by ID or name (partial or full name). If the part or parts are found, the application highlights a single part or filters multiple parts. If the part or parts are not found, the application displays an error message in the UI or in a dialog box.

• If the search field is set to empty, the table should be repopulated with all available parts.

• The top table should be identical to the Parts TableView in the Main form.

• The user can select a part from the top table. The user then clicks the Add button, and the part is copied to the bottom table. (This associates one or more parts with a product.)

• The Remove Associated Part button removes a selected part from the bottom table. (This dissociates or removes a part from a product.)

• After saving the data, the user is automatically redirected to the Main form.

• Canceling or exiting this form redirects users to the Main form.

Note: When a product is deleted, so can its associated parts without affecting the part inventory. The Remove Associated Part button removes a selected part from the bottom table. (This dissociates or removes a part from a product.)

2. The Modify Product form
    - The text fields populate with the data from the chosen product, and the bottom TableView populates with the associated parts.
    - The user can search for parts (top table) by ID or name (partial or full name). If the part or parts are found, the application highlights a single part or filters multiple parts. If the part is not found, the application displays an error message in the UI or a dialog box.
    - If the search text field is set to empty, the table should be repopulated with all available parts.
    - The top table should be identical to the Parts TableView in the Main form.
    - The user may modify or change data values.
        - The product ID text field must be disabled and cannot be edited or changed.
    - The user can select a part from the top table. The user then clicks the Add button, and the part is copied to the bottom table. (This associates one or more parts with a product.)
    - The user may associate zero, one, or more parts with a product.
    - The user may remove or disassociate a part from a product.
    - After saving modifications to the product, the user is automatically redirected to the Main form.
    - Canceling or exiting this form redirects users to the Main form.

    Note: The Remove Associated Part button removes a selected part from the bottom table. (This dissociates or removes a part from a product.)

G. Write code to implement input validation and logical error checks using a dialog box or message in the UI displaying a descriptive error message for *each* of the following circumstances:

- Min should be less than Max; and Inv should be between those two values.
- The user should not delete a product that has a part associated with it.
- The application confirms the “Delete” and “Remove” actions.
- The application will not crash when inappropriate user data is entered in the forms; instead, error messages should be generated.

H. Provide a folder containing Javadoc files that were generated from the IDE or via the command prompt from part B. In a comment above the main method header declaration, please specify where this folder is located.

I. Demonstrate professional communication in the content and presentation of your submission.