# QKM2 — QKM2 TASK 1: OBJECT-ORIENTED APPLICATION DEVELOPMENT

SOFTWARE I — C482
PRFA — QKM2

TASK OVERVIEW SUBMISSIONS EVALUATION REPORT

## COMPETENCIES

**430.02.05 : Classes and Interfaces**
The graduate designs software solutions with appropriate classes, objects, methods, and interfaces to achieve specific goals.

**430.02.06 : Object-Oriented Principles**
The graduate implements object-oriented design principles (e.g., inheritance, encapsulation, and abstraction) in developing applications for ensuring the application's scalability.

**430.02.07 : Application Development**
The graduate produces applications using Java programming language constructs to meet business requirements.

**430.02.08 : Exception Handling**
The graduate incorporates simple exception handling in application development for improving user experience and application stability.

**430.02.09 : User Interface Development**
The graduate develops user interfaces to meet project requirements.

## INTRODUCTION

Throughout your career in software design and development, you will be asked to create applications with various features and functionality based on business requirements. For this assessment, you will create a Java desktop application using the solution statements provided in the requirements section of this assessment.

The skills you showcase in your completed application will be useful in responding to technical interview questions for future employment. This application may also be added to your portfolio to show to future employers.

The preferred integrated development environment (IDE) for this assignment is NetBeans version 11.1 or later or IntelliJ IDEA (Community Edition). Use the links in the web links section of this assessment to install one of these IDEs. If you choose to use another IDE, you must export your project into NetBeans 11.1 or later or IntelliJ IDEA format or your submission will be returned.

This assessment also requires the following software: JDK 17 (LTS) and JavaFX SDK or Module (for NetBeans or IntelliJ IDEA), and Scene Builder, which are also available for download in the web links section of this assessment, as well as a video demonstration of the completed application.

Your submission should include a zip file with all the necessary code files to compile, support, and run your application. Your submission should also include a folder with descriptive Javadoc comments in the .java files. The zip file submission must keep the project file and folder structure intact for the IDE.

In NetBeans, zip your file by going to File > Export Project > To ZIP and click Export. In IntelliJ IDEA, go to File > Export to Zip File and click OK. If you try to zip your project files with an external program, it will include the build files and make the zip files too large for submission.

*Note: You may receive an error message upon submitting your files because the automated plagiarism detectors will not be able to access the zipped file, but the evaluation team members will run their checks manually when evaluating your submission.*

# 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

*Your submission must be your original work. No more than a combined total of 30% of the submission and no more than a 10% match to any one individual source can be directly quoted or closely paraphrased from sources, even if cited correctly. The originality report that is provided when you submit your task can be used as a guide.*

*You must use the rubric to direct the creation of your submission because it provides detailed criteria that will be used to evaluate your work. Each requirement below may be evaluated by more than one rubric aspect. The rubric aspect titles may contain hyperlinks to relevant portions of the course.*

*Tasks may **not** be submitted as cloud links, such as links to Google Docs, Google Slides, OneDrive, etc., unless specified in the task requirements. All other submissions must be file types that are uploaded and submitted as attachments (e.g., .docx, .pdf, .ppt).*

**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.

## File Restrictions

File name may contain only letters, numbers, spaces, and these symbols: ! - _ . * ' ( )
File size limit: 400 MB
File types allowed: doc, docx, rtf, xls, xlsx, ppt, pptx, odt, pdf, txt, qt, mov, mpg, avi, mp3, wav, mp4, wma, flv, asf, mpeg, wmv, m4v, svg, tif, tiff, jpeg, jpg, gif, png, zip, rar, tar, 7z

# RUBRIC

**A1:MAIN FORM**

| NOT EVIDENT | APPROACHING COMPETENCE | COMPETENT |
|---|---|---|
| No code is provided for the Main form. | The submitted Main form does not closely match the organization of the GUI layout or does not include *all* required components. Or the code uses a prohibited tool to create the GUI, contains errors, or is incomplete. | The submitted Main form closely matches the organization of the GUI layout and includes *all* required components. The code uses a permitted tool to create the GUI, does not contain errors, and is complete. |

**A2:ADD PART FORM**

| NOT EVIDENT | APPROACHING COMPETENCE | COMPETENT |
|---|---|---|
| No code is provided for the Add Part form. | The submitted Add Part form does not closely match the organization of the GUI layout or does not include *all* required components. Or the code uses a prohibited tool to create the GUI, contains errors, or is incomplete. | The submitted Add Part form closely matches the organization of the GUI layout and includes *all* required components. The code uses a permitted tool to create the GUI, does not contain errors, and is complete. |

**A3:MODIFY PART FORM**

| NOT EVIDENT | APPROACHING COMPETENCE | COMPETENT |
|---|---|---|
| No code is provided for the Modify Part form. | The submitted Modify Part form does not closely match the organization of the GUI layout or does not include *all* required components. Or the code uses a prohibited tool to create the GUI, contains errors, or is incomplete. | The submitted Modify Part form closely matches the organization of the GUI layout and includes *all* required components. The code uses a permitted tool to create the GUI, does not contain errors, and is complete. |

**A4:ADD PRODUCT FORM**

| NOT EVIDENT | APPROACHING COMPETENCE | COMPETENT |
|---|---|---|

No code is provided for the Add Product form.

The submitted Add Product form does not closely match the organization of the GUI layout or does not include *all* required components. Or the code uses a prohibited tool to create the GUI, contains errors, or is incomplete.

The submitted Add Product form closely matches the organization of the GUI layout and includes *all* required components. The code uses a permitted tool to create the GUI, does not contain errors, and is complete.

A5:MODIFY PRODUCT FORM

**NOT EVIDENT**

No code is provided for the Modify Product form.

**APPROACHING COMPETENCE**

The Modify Product form does not closely match the organization of the GUI layout or does not include *all* required components. Or the code uses a prohibited tool to create the GUI, contains errors, or is incomplete.

**COMPETENT**

The Modify Product form closely matches the organization of the GUI layout and includes *all* required components. The code uses a permitted tool to create the GUI, does not contain errors, and is complete.

B:JAVADOC COMMENTS

**NOT EVIDENT**

Javadoc comments are not provided.

**APPROACHING COMPETENCE**

Javadoc comments are provided but not for *each* class member, or the comments do not appear throughout the code. Or the comments do not include detailed descriptions of a corrected logical or runtime error or how the error was corrected in the code, or a description of a future enhancement that would extend the functionality of the application if it were to be updated.

**COMPETENT**

Javadoc comments are provided for *each* class member throughout the code and include detailed descriptions of a logical or runtime error corrected in the code, how the error was corrected, and a description of a future enhancement that would extend the functionality of the application if it were to be updated.

C:CLASS STRUCTURE

**NOT EVIDENT**

No class code is provided.

**APPROACHING COMPETENCE**

**COMPETENT**

The application maps to the UML class diagram and includes *all*

| | | |
|---|---|---|
| | The application maps to the UML class diagram but does not include *all* classes and members shown in the diagram. Or the code does not correctly demonstrate *each* given point or does not include the unmodified Part class that is provided. | classes and members shown in the diagram. The code correctly demonstrates *each* given point, and includes the unmodified Part class that is provided. |

**D1:PARTS PANE**

| NOT EVIDENT | APPROACHING COMPETENCE | COMPETENT |
|---|---|---|
| A Parts pane is not included on the Main form. | One or more buttons on the Parts pane do not function properly. Or the application does not display matching search results in the Parts TableView if the part or parts are found, or the application does not display an error message in the UI or in a dialog box if the part or parts are not found. Or the empty search field functions incorrectly. | *Each* button on the Parts pane functions properly. The application displays matching search results in the Parts TableView if the part or parts are found, or the application displays an error message in the UI or in a dialog box if the part or parts are not found. The empty search field functions correctly. |

**D2:PRODUCTS PANE**

| NOT EVIDENT | APPROACHING COMPETENCE | COMPETENT |
|---|---|---|
| A Products pane is not included on the Main form. | 1 or more buttons on the Products pane do not function properly. Or the application does not display matching search results in the Products TableView if the product or products are found, or the application does not display an error message in the UI or in a dialog box if the product or products are not found. Or the empty search field functions incorrectly. | *Each* button on the Products pane functions properly. The application displays matching search results in the Products TableView if the product or products are found, or the application displays an error message in the UI or in a dialog box if the product or products are not found. The empty search field functions correctly. |

**D3:EXIT BUTTON**

| NOT EVIDENT | APPROACHING COMPETENCE | COMPETENT |
| --- | --- | --- |
| The Exit button does not close the application. | Not applicable. | The Exit button functions properly by closing the application. |

E1:ADD PART FORM FUNCTIONALITY

| NOT EVIDENT | APPROACHING COMPETENCE | COMPETENT |
| --- | --- | --- |
| The Add Part form functionality is not included. | The Add Part form does not include correctly functioning In-House or Outsourced radio buttons, does not auto-generate a unique disabled part ID text field, or does not allow users to enter the given part details into their corresponding active text fields. Or users are not automatically redirected to the Main form after saving data or after canceling or exiting the form. | The Add Part form includes correctly functioning In-House and Outsourced radio buttons, auto-generates a unique part ID but leaves the part ID text field disabled, and allows users to enter the given part details into their corresponding active text fields. Users are automatically redirected to the Main form after saving data or after canceling or exiting the form. |

E2:MODIFY PART FORM FUNCTIONALITY

| NOT EVIDENT | APPROACHING COMPETENCE | COMPETENT |
| --- | --- | --- |
| The Modify Part form functionality is not included. | The Modify Part form does not include populated text fields with the data from the chosen part. Or the form does not include correctly functioning In-House or Outsourced radio buttons or does not retain the part ID when the Save button is clicked. Or users can modify the part ID, cannot modify existing values in the text fields, or are not automatically redirected to the Main form after saving modifications to the part or after canceling or exiting the form. | The Modify Part form includes populated text fields with the data from the chosen part. The form includes correctly functioning In-House or Outsourced radio buttons and retains the part ID when the Save button is clicked. Users can modify existing values in the text fields except the part ID and are automatically redirected to the Main form after saving modifications to the part or after canceling or exiting the form. |

F1:ADD PRODUCT FORM FUNCTIONALITY

| NOT EVIDENT | APPROACHING COMPETENCE | COMPETENT |
| --- | --- | --- |
| The Add Product form functionality is not included. | The Add Product form does not auto-generate a unique product ID. Or the auto-generated product ID does not populate or is not disabled. Or the user cannot enter the given product details or search for parts by ID or name. Or the application does not display matching search results for a single part or does not filter multiple parts if they are found, or the application does not display an error message in the UI or a dialog box if the part or parts are not found. Or the empty search text field does not function correctly, the top table is not identical to the Parts TableView in the Main form, the Add Button does not copy the selected part to the lower table, the Remove Associated Part button does not function as required, or users are not automatically redirected to the Main form after saving modifications to the part or after canceling or exiting the form. | The Add Product form auto-generates a unique product ID. The auto-generated product ID populates but is disabled. The user can enter the given product details and search for parts by ID or name. The application displays matching search results for a single part or filters multiple parts if they are found, or the application displays an error message in the UI or a dialog box if the part or parts are not found. The empty search text field functions correctly, the top table is identical to the Parts TableView in the Main form, the Add button copies the selected part to the lower table, the Remove Associated Part button functions as required, and users are automatically redirected to the Main form after saving modifications to the part or after canceling or exiting the form. |

F2:MODIFY PRODUCT FORM FUNCTIONALITY

| NOT EVIDENT | APPROACHING COMPETENCE | COMPETENT |
| --- | --- | --- |
| The Modify Product form functionality is not included. | The Modify Product form does not include text fields populated with data from the chosen product, or the bottom TableView does not populate with the associated parts. Or the form does not allow the user to search for | The Modify Product form includes text fields populated with data from the chosen product, and the bottom TableView populates with the associated parts. The form allows the user to search for parts by ID or name and displays matching search re- |

| | | |
|---|---|---|
| | parts by ID or name or does not display matching search results for a single part or does not filter multiple parts if they are found or does not display an error message in the UI or in a dialog box if the part or parts are not found. The empty search text field does not function correctly. The user cannot modify data values, or the product ID is enabled. Or the user cannot associate, remove, or disassociate a part from a product. Users are not automatically redirected to the Main form after saving modifications to the part or after canceling or exiting the form. | sults for a single part or filters multiple parts if they are found, or it displays an error message in the UI or in a dialog box if the part or parts are not found. The empty search text field functions correctly, the top table is identical to the Parts TableView in the Main form. The user can modify data values, and the product ID is disabled. The user can associate, remove, or disassociate a part from a product. Associated parts are selected from the top table and are stored on the bottom table. Users are automatically redirected to the Main form after saving modifications to the part or after canceling or exiting the form. |

**G:INPUT VALIDATION AND LOGICAL ERROR CHECKS**

| NOT EVIDENT | APPROACHING COMPETENCE | COMPETENT |
|---|---|---|
| No code is provided. | The code does not implement input validation or logical error checks for *each* of the given circumstances or does not display a descriptive error message when an error is detected. | The code implements input validation and logical error checks for *each* of the given circumstances and displays a descriptive error message when an error is detected. |

**H:JAVADOC FOLDER**

| NOT EVIDENT | APPROACHING COMPETENCE | COMPETENT |
|---|---|---|
| The submission does not include a folder containing Javadoc files or does not specify, in a comment above the main method header declaration, where the folder is located. | The submission includes a folder containing some Javadoc files but not all, or the files were not generated from the IDE or via the command prompt from part B. | The submission includes a folder containing Javadoc files that were generated from the IDE or via the command prompt from part B and specifies, in a comment above the main method header declaration, where the folder is located. |

I:PROFESSIONAL COMMUNICATION

| NOT EVIDENT | APPROACHING COMPETENCE | COMPETENT |
| --- | --- | --- |
| Content is unstructured, is disjointed, or contains pervasive errors in mechanics, usage, or grammar. Vocabulary or tone is unprofessional or distracts from the topic. | Content is poorly organized, is difficult to follow, or contains errors in mechanics, usage, or grammar that cause confusion. Terminology is misused or ineffective. | Content reflects attention to detail, is organized, and focuses on the main ideas as prescribed in the task or chosen by the candidate. Terminology is pertinent, is used correctly, and effectively conveys the intended meaning. Mechanics, usage, and grammar promote accurate interpretation and understanding. |

## WEB LINKS

- C482 Application Video Demonstration
- IntelliJ IDEA (Community Edition) IDE
- JavaFX SDK (Windows SDK or Mac OS X SDK)
- JDK 11 (LTS) (Windows x64 installer or macOS Installer)
- NetBeans 11 or later
- Scene Builder for Java 17
- JDK 17 (LTS) (Windows x64 installer or macOS Installer)

## SUPPORTING DOCUMENTS

- Software 1 GUI Mockup.docx
- Part.java
- UML Class Diagram.pdf