

# **Assignment Tutorial Letter 2025**

**Advanced Programming**  
**COS3711**

**Year module**

**Computer Science Department**

Assignment 2 - Questions

BARCODE

## Assignment 2

### 1. Introduction

Please read this whole assignment tutorial letter before starting to ensure that you know what is expected of you and you do not get surprised by some requirement later in the development process.

Use `CMake` to set up the projects and use `QMainWindow` instances for the user interfaces so that you can implement the functionality provided by a `QMainWindow` (such as menus, a toolbar, and a status bar) to meet the requirements of the scenario given below. Please note that you must not use Qt Designer, and you are expected to manually set up GUIs to ensure that you properly handle memory using Qt's parent-child functionality.

Marks will also be awarded for following good programming practice.

- Follow standard naming conventions: class names start with a capital letter, variable and function names start with a lowercase letter, using camelCase for names made up of multiple words.
- Ensure consistent code layout and use of blank lines.
- Use forward class declarations in header files.
- Use initialiser lists in constructors.
- Have proper GUI management: setting cursor focus, sequential tabbing, clearing input widgets (like text input fields being cleared and spin boxes being returned to some default value), and enabling and disabling buttons/menu items as appropriate.
- Provide appropriate user feedback.
- Your code should build and run without any warnings.

Consider the following scenario and then design a solution to meet the requirements listed.

### 2. Scenario

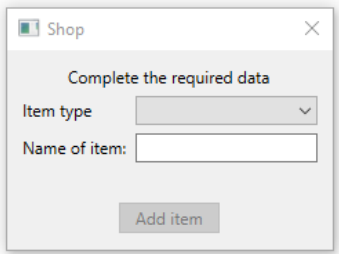
Implement an application that tracks items bought from a store. Screenshots of possible interfaces have been provided as guidance below.

#### *Store application*

The store keeps a list of customers so that when something is purchased from the store, it is recorded against that customer's name. Only the customer's name is required.

Only two items are currently sold in the store currently – books and magazines, and only the name of the item is required. Clearly, a list of such items is needed, and the user should be able to add items to the list. When an item is added, the application should automatically make a backup in

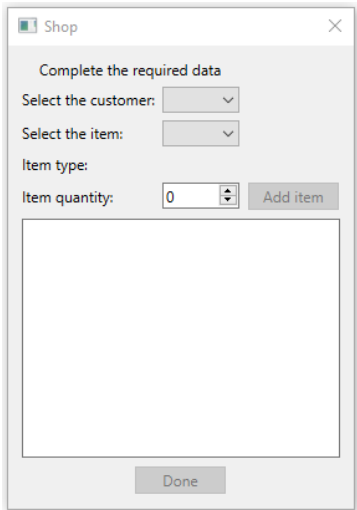
memory in case something goes wrong when the application is being used. Provision should thus be made to restore this list when necessary.



A dialog box titled "Shop" with a close button (X). It contains a section "Complete the required data" with two input fields: "Item type" (a dropdown menu) and "Name of item:" (a text box). Below these fields is an "Add item" button.

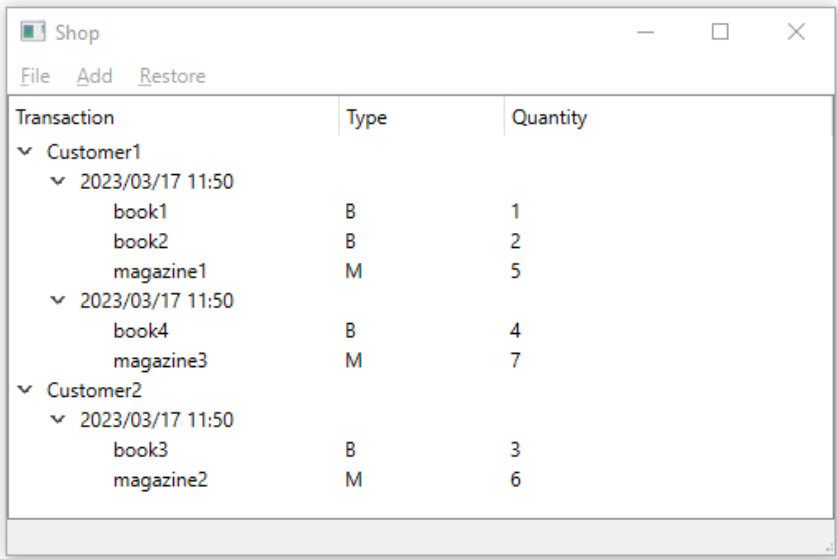
Clearly, for the sake of data integrity, you do not want the user to create multiple copies of these lists.

When a customer purchases items, a transaction is recorded. The date/time of the purchase is noted, as is the name, type, and quantity of each item purchased as part of the transaction. Use an appropriate widget to indicate which items have already been added as part of this transaction.



A dialog box titled "Shop" with a close button (X). It contains a section "Complete the required data" with three input fields: "Select the customer:" (a dropdown menu), "Select the item:" (a dropdown menu), and "Item type:" (a dropdown menu). Below these fields is an "Item quantity:" field with a numeric input (0) and a spin button. To the right of the quantity field is an "Add item" button. At the bottom of the dialog is a "Done" button.

All transactions should be displayed on the main GUI. A tree model (and appropriate view) should be used so that a user can see a customer's transactions grouped together.



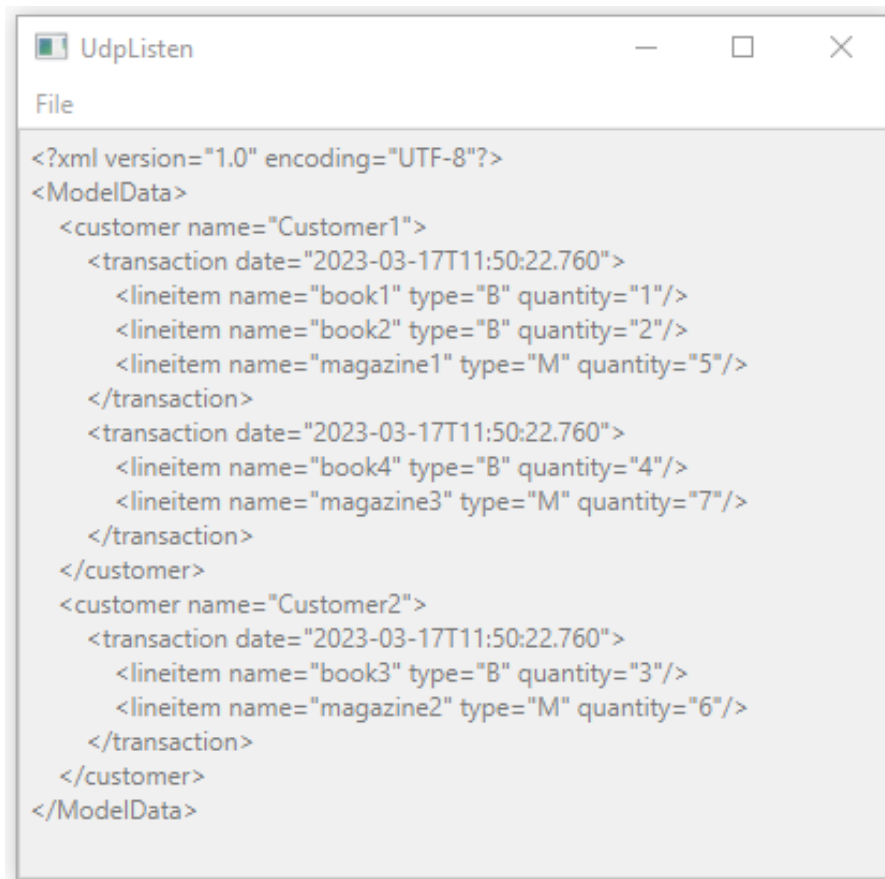
A screenshot of the "Shop" application window. It has a menu bar with "File", "Add", and "Restore". The main area displays a tree view of transactions. The tree structure is as follows:

Transaction	Type	Quantity
Customer1		
2023/03/17 11:50		
book1	B	1
book2	B	2
magazine1	M	5
2023/03/17 11:50		
book4	B	4
magazine3	M	7
Customer2		
2023/03/17 11:50		
book3	B	3
magazine2	M	6

The user should be able to broadcast (using UDP) the contents of the model in XML format. This task should be run as a thread in the main application. The required XML format can be seen in the image in the next section.

### *UDP receiver application*

Create a separate application that simply listens for the broadcast message and displays the received data (in XML format) on the GUI.



## 3. Requirements

The following general requirements should be noted.

- Follow good OOP design principles.
- You should use menus in your application.
- Pointers should be used for all instances of objects, and memory should be properly managed.
- Appropriate design patterns should be used where sensible.

## 4. Extras

Bonus marks will be awarded for the following.

- The data members required for the classes are very basic and can be extended (like adding a price for each item and tracking the number of items available).

- Using `QMainWindow` functionality:
  - Splash screen
  - Application icon
  - Toolbar
  - About and Help

## 5. Submission

Please check the *How to submit the assessments* page in the *Module orientation* lesson on myUnisa before submitting this assignment for information about how to submit the assignment.

- Use `CMake` when setting up your assignment.
- Submit only the project folders.
- You should not submit the build-desktop folders.
- Zip the two folders (for the two parts of the project) into one zipped file for submission.

© Unisa  
2025