

Department of Computing and Mathematics

ASSIGNMENT COVER SHEET

Unit code and title:	6G7V0039 – Advanced Object Oriented Programming
Credit value:	30
Assignment set by:	Darren Dancey
Assignment ID: (e.g. 1CWK50)	1CWK100
Assignment weighting:	100
Assignment title:	The Everything Store
Type: (Group/Individual)	Individual
Hand-in deadline:	
Hand-in format and mechanism:	Moodle Zip file (including Checklist, and code or repl.it link)

Learning outcomes being assessed:

LO1: Apply computational thinking and fundamental programming techniques to solve problems.
 LO2: Use object-oriented programming techniques in the design and implementation of software.
 LO3: Adopt a reasoned approach to identify and rectify software defects in their programs.
 LO4: Use standard software testing techniques to verify and demonstrate the correctness of their code.

Note: it is your responsibility to make sure that your work is complete and available for marking by the deadline. Make sure that you have followed the submission instructions carefully, and your work is submitted in the correct format, using the correct hand-in mechanism (e.g. Moodle upload). If submitting via Moodle, you are advised to check your work after upload, to make sure it has uploaded properly. Do not alter your work after the deadline. You should make at least one full backup copy of your work.

Penalties for late hand-in: see Assessment Regulations for Undergraduate/Postgraduate Programmes of Study on the [Student Life web pages](#). The timeliness of submissions is strictly monitored and enforced.

All coursework has a late submission window of 7 days (i.e. 5 working days), but any work submitted within the late window will be capped at 50%, unless you have an agreed extension. Work submitted after the 7-day late window will be capped at zero, unless you have an agreed extension. See below for further information on extensions.

Please note that individual tutors are unable to grant extensions to coursework.

Extensions: For most coursework assessments, you can request a 7-day extension through Moodle. This will not apply to in-class tests, presentations, interviews, etc, that take place at a specific time. If you need a longer extension you can apply for an Evidenced Extension. For an Evidenced Extension you **MUST** submit 3rd party evidence of the condition or situation which has negatively impacted on your ability to

submit or perform in an assessment.

Plagiarism: Plagiarism is the unacknowledged representation of another person's work, or use of their ideas, as one's own. Manchester Metropolitan University takes care to detect plagiarism, employs plagiarism detection software, and imposes severe penalties, as outlined in the Student Handbook (http://www.mmu.ac.uk/academic/casqe/regulations/docs/policies_regulations.pdf and Regulations for Undergraduate Programmes (<https://www.mmu.ac.uk/academic/casqe/regulations/assessment/docs/ug-regs.pdf>). Bad referencing or submitting the wrong assignment may still be treated as plagiarism. If in doubt, seek advice from your tutor.

If you are unable to upload your work to Moodle: If you have problems submitting your work through Moodle you can email it to the Assessment Team's Contingency Submission Inbox using the email address submit@mmu.ac.uk. You should say in your email which unit the work is for, and ideally provide the name of the Unit Leader. The Assessment team will then forward your work to the appropriate person. If you use this submission method, your work must be emailed by the published deadline, or it will be logged as a late submission. Alternatively, you can save your work to your university OneDrive and submit a Word document to Moodle which includes a link to the folder. It is your responsibility to make sure you share the OneDrive folder with the Unit Leader.

As part of a plagiarism check, you may be asked to attend a meeting with the Unit Leader, or another member of the unit delivery team, where you will be asked to explain your work (e.g. explain the code in a programming assignment). If you are called to one of these meetings, it is very important that you attend.

Assessment Criteria:	Indicated in the attached assignment specification.
Formative Feedback:	<i>Formative feedback based on lab exercises designed to replicate core aspects of the assignment</i>
Summative Feedback Format:	<i>Feedback on Moodle using marking grid and brief overall comments on aspects done well and aspects to improve</i>

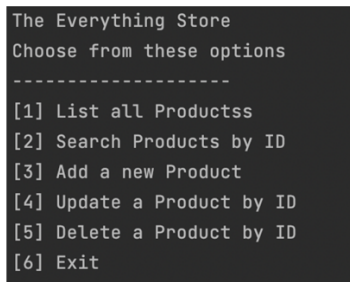
Aim

This unit covers programming fundamentals, object-oriented programming principles and practice.

The aim of this assignment is to develop a web-based stock control and ordering system. The system will allow a product's details (SKU, description, price, etc.) to be added and removed from a SQLite DB. The system will also have a list of customers with names and addresses stored for each customer. An order will have a single customer from the customer list and list of products being ordered. The system will be able to calculate the total price of an order when displaying that order's summary. Advanced versions of the system should be able to record the amount of stock being held of each product and update stock levels as orders are created and deleted.

The Brief

The first stage of the assignment requires that you develop a menu-based console application see a partial example below (Fig. 1)



```
The Everything Store
Choose from these options
-----
[1] List all Productss
[2] Search Products by ID
[3] Add a new Product
[4] Update a Product by ID
[5] Delete a Product by ID
[6] Exit
```

Figure 1: Stock control inventory menu

Specifically, your menu should provide options to

- retrieve all products, customers,
- search for a specific product, customer,
- add a new product into the system,
- update the details of an existing customer, product
- delete product and customers.

In addition to the above your code should be of high quality and robust.

The following provides a step-by-step guide to the assignment but you can tackle it in your own way if you so wish.

Step 1: Create the Product class

The first step is to create a product class which store the required information about a product and maps to the data stored in the everythingstore.sqlite database. The database has a table called product which store the product details as show below. The Product.java class models the data entities in the database. Consequently, it needs the following instance variables to be defined.

```
public class Product {
    int id;
    String SKU; //stock keeping unit (a unique code for each product)
    String description;
```

```
String category;
int price;
}
```

You need to create a constructor to allow these instance variables to be initialised which has the form shown below.

```
public Product(String SKU, String description,String category int price) {
    // insert code here to initialise instance variables
}
```

You should also create getter and setter methods for the instance variables together with a toString() method.

Step 2: Step 2: Create the ProductDAO class

The next step is to develop the ProductDAO class using the Java Database Connectivity (JDBC) API . This is the data access object (DAO) that provides an interface to the SQLite database and should contain the create, read, update and delete (CRUD) methods. The basic structure of the ProductDAO class is shown in Fig. 3 and should have methods for each of the CRUD operations.

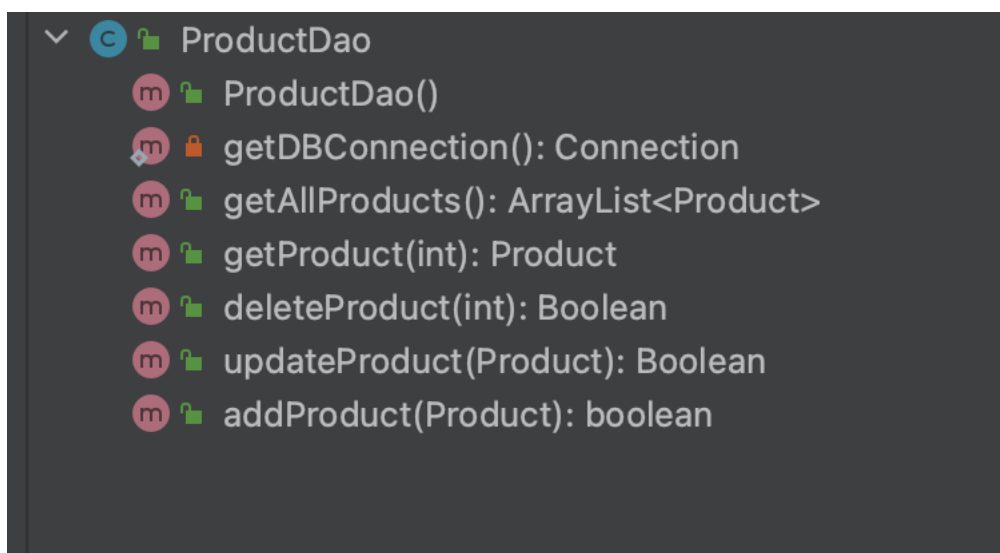


Fig. 3. ProductDAO class

The getAllProducts() method uses the SQL SELECT statement to retrieve a product records from the database as shown below.

```
String query = "SELECT * FROM product;";
```

For the getProduct(int id) method you need to specify a criterion for the data to be retrieved using the SQL WHERE clause. In the example below the id number is used in the SQL SELECT statement.

```
String query = "SELECT * FROM product WHERE ID =" + id + ";;";
```

Similarly for the deleteProduct(int id) method you will need to specify a criterion for the record to be deleted. In the example below the Product ID number is used.

```
String query = "DELETE FROM product WHERE ID = " + id + ";;";
```

The insertProduct method will need to use the SQL statement INSERT INTO product(...) VALUES (...) and the updateProduct() method will need to use SQL UPDATE statement.

The w3Schools tutorial provides a refresher on SQL statements should you need it.

<https://www.w3schools.com/sql/default.asp>

You will need to add further methods to the Product.java class which will be needed when implementing additional features. You will receive additional marks if your ProductDAO uses SQL prepared statements (i.e. pre-compiled SQL statements with parameter placeholders). They have the following benefits. They help in preventing SQL injection attacks. They make it easier to set SQL parameters. Performance may be improved as statements are pre-compiled.

Step 3: Testing CRUD operation with console menu system

This stage of the assignment requires that you write a Controller class with a menu system to test each of the CRUD (i.e. Create, Retrieve, Update and Delete) database operations developed for the ProductDAO class to check their functionality. You will need to use the Scanner class to receive input from the keyboard.

```
Scanner input = new Scanner(System.in);
```

Your menu system should look like that below so that when, for example, option 1 is selected all products stored in the database are displayed as shown below.

```
String selection;
```

```
do {
    System.out.println("-----");
    System.out.println("The Everything Store");
    System.out.println("Choose from these options");
    System.out.println("-----");

    System.out.println("[1] List all Products");
    // insert code to print remaining options
    System.out.println("[99] Exit");
    System.out.println();

    selection = in.nextLine();

    switch (selection) {
        case "1":

            for (int i = 0; i < products.size(); i++){
                System.out.println(products.get(i));
            }
            System.out.println();
            break;
```

```
case "2":  
    System.out.println("\nSearch for a product by SKU");  
    String SKU = in.nextLine();  
    // code to find and display product based on SKU  
    break;  
} while (!selection.equals("99"));
```

Step 4: Implement the options to find, add and modify products held within the system. The system should prompt the user for any required information as shown in Figure 2.

```

The Everything Store
Choose from these options
-----
[1] List all Productss
[2] Search Products by ID
[3] Add a new Product
[4] Update a Product by ID
[5] Delete a Product by ID
[6] Exit

Update a new Product

Enter Product ID to update
Product{id=4, SKU='PSX126', description='Playstation 5 Digital Edition', price=49999, category='Gaming'}
Updating Product with ID:4
Please enter SKU

Please enter new Description

Please enter new Category

Please enter price
39900
-----
The Everything Store
Choose from these options
-----
[1] List all Productss
[2] Search Products by ID
[3] Add a new Product
[4] Update a Product by ID
[5] Delete a Product by ID
[6] Exit

Retrieving all Products ...
Product{id=1, SKU='MS123', description='Xbox Series X', price=44999, category='Gaming'}
Product{id=2, SKU='MS124', description='Xbox Series S', price=24999, category='Gaming'}
Product{id=3, SKU='PSX125', description='Playstation 5', price=44999, category='Gaming'}
Product{id=4, SKU='PSX126', description='Playstation 5 Digital Edition', price=39900, category='Gaming'}
Product{id=5, SKU='PEN001', description='A Brief History of Time', price=1099, category='Book'}
Product{id=6, SKU='PEN002', description='Generation X', price=599, category='Book'}
Product{id=33, SKU='PEN232', description='Atlas Shrugged', price=499, category='Book'}

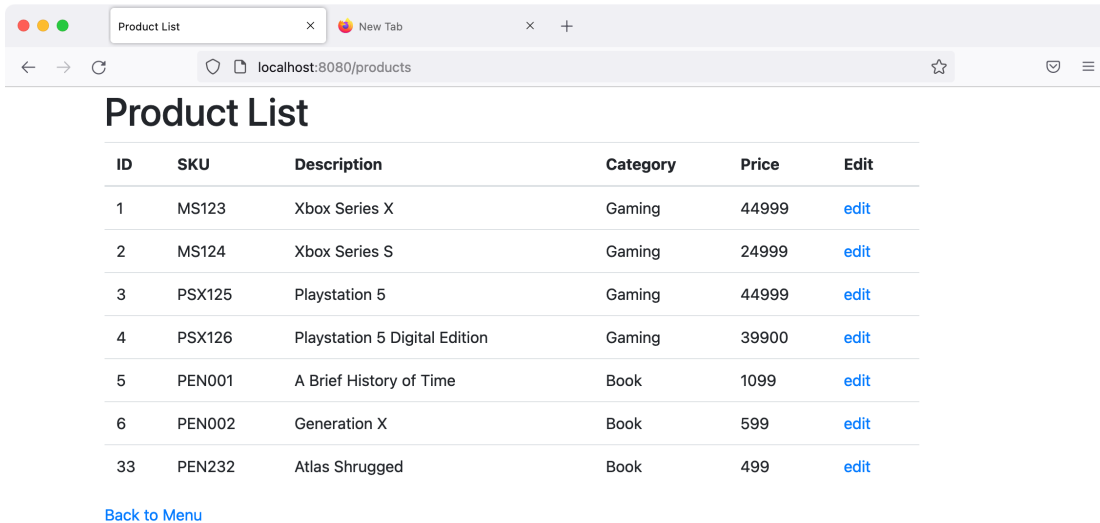
-----
The Everything Store
Choose from these options
-----
[1] List all Productss
[2] Search Products by ID
[3] Add a new Product
[4] Update a Product by ID
[5] Delete a Product by ID
[6] Exit

```

Figure 2: Example Menu Interaction

Step 5: Web Front-end (Displaying Products)

This stage of the assignment requires that you write server code which displays products stored in the database on a web page as shown in Figure 3.



ID	SKU	Description	Category	Price	Edit
1	MS123	Xbox Series X	Gaming	44999	edit
2	MS124	Xbox Series S	Gaming	24999	edit
3	PSX125	Playstation 5	Gaming	44999	edit
4	PSX126	Playstation 5 Digital Edition	Gaming	39900	edit
5	PEN001	A Brief History of Time	Book	1099	edit
6	PEN002	Generation X	Book	599	edit
33	PEN232	Atlas Shrugged	Book	499	edit

[Back to Menu](#)

Figure 3: Displaying Products using HTML Table

Step 6: Repeat the steps 2-5 so that customers can be added to the system using both the console menu and the web interface.

A customer needs to have at least the following fields

```
int customerID;  
String firstName;  
String secondName;  
Address address;  
String telephoneNumber;
```

Address is also a new class with the following fields

```
String house;  
String addressLine1;  
String addressLine2;  
String country;  
String postCode;
```

Step 7: Administration Login/Logout

This stage of the assignment requires that you use the “users” table in the everythingstore.sqlite database to provide a login system to allow an administrator to log into the system to insert, edit update and delete products stored in the database as shown in Figure 5. For the purposes of this assignment this is a proof of concept or mock up and is not expected to be secure.

Username:

Password:

[Login](#)

Figure 4: Admin Interface

Step 7: Web Interface (Administrators)

This stage of the assignment requires that you develop code to allow a user to insert, edit, update and delete products and customers stored in the database as shown in Figure 5.

ID	SKU	Description	Category	Price	Edit	Delete
1	MS123	Xbox Series X	Gaming	44999	edit	delete
2	MS124	Xbox Series S	Gaming	24999	edit	delete
3	PSX125	Playstation 5	Gaming	44999	edit	delete
4	PSX126	Playstation 5 Digital Edition	Gaming	39900	edit	delete
5	PEN001	A Brief History of Time	Book	1099	edit	delete
6	PEN002	Generation X	Book	599	edit	delete
33	PEN232	Atlas Shrugged	Book	499	edit	delete

[Back to Menu](#)

Figure 5: Product List with Edit and Delete Options

Step 8: Filters and Search

Implement a filter on the product list so only products of a certain category will be shown. Implement a search function that finds products which contain the search string with the product description.

Step 9: Documentation and Testing

Document the code using JavaDoc and produce a documentation set using JavaDoc program. Using Junit create a test suite for your Customer and Product classes.

Step 10: Advanced Coding Approaches

Advanced use of Java such as using appropriate design patterns and identifying the usage within the code or using lambda expressions appropriately within the code.

Step 11: Advanced Features (required for Distinction)

You will be rewarded for implementing additional features which sensibly enhance the system. In your assignment template form please specify any additional features that you have added to the system.

Some suggested features are below:

Implement Password hashing. Ideally plaintext passwords should not be stored in a database. Use a hashing function such as MD5 to store only hashed passwords in the database.

Implement a web shopping basket type feature that allows users to create an order with several items. The basket can show the total value of the items stored.

The system currently only has one type of product with a simple description. Expand the type of products that can be stored with unique attributes stored for each product type. For example, Books would be a type of product with the additional attributes of Author and publisher, DVD would be a type of product with additional attributes of title and director.

Hand In:

You are required to up-load your **project files or a repl.it link** and a **completed assignment template form** (see below) indicating which features have been implemented successfully. Please ensure that all code submitted compiles and runs at the command line and your name and student number are included as comments in your source code files.

Assignment Template Form

Assignment Functionality Completed:

Product class implemented

- ☐ Product Class fully implemented and tested

Customer class implemented

- ☐ Customer Class fully implemented and tested including the address class

Menu and Database Functionality

- ☐ Menu option to list all customers in the system
- ☐ Menu option to add a customer to the system
- ☐ Menu option to find a customer by ID
- ☐ Menu option to update a customer
- ☐ Menu option to delete a customer from the system
- ☐ Menu option to list all products in the system
- ☐ Menu option to add a product to the system
- ☐ Menu option to find a product by ID
- ☐ Menu option to update a product by ID
- ☐ Menu option to delete a product by ID

Web Functionality

- ☐ Web page to list all customers in the system
- ☐ Web function to add a customer to the system

- ☐ Web function to edit a customer in the system
- ☐ Web function to delete a customer from the system
- ☐ Web function to list all products in the system
- ☐ Web function on to add and edit a product in the system
- ☐ Web function delete a product

Advanced Features

- ☐ Shopping Basket
- ☐ Multiple Product types
- ☐ Search descriptions and/or filter for products
- ☐ Lambda expressions
- ☐ Unit Testing
- ☐ JavaDoc
- ☐ Design Pattern(s)

Marking Scheme

Product Class	Marks	Databases Connectivity		Web Front End	Marks	Customer Functionality	Marks	Advanced Functionality	Marks
Class with appropriate fields, getters and setters and constructor	5	Connection to database and CRUD method to return all Products	5	Server connects to the SQLite database and allows books to be displayed using an HTML table	5	A complete customer class with address	5	Appropriate Advanced functionality such as code to implement a realistic basket function or product types, filter or search	20
Console Menu Testing		CRUD method to return single product based on ID	5	The web front-end provides functionality to add, edit and delete products stored in the database with password hashing	10	CRUD database methods to support customer	5	Code quality and appropriate advanced code practices used such as Design Patterns used and identified	5
Menu System that allows all books to be displayed and a single Product to be displayed based on a search by ID Menu Allows Products to be added, updated and deleted	5	CRUD methods to insert and delete products	5	Functionality to allow only admin users to modify the database through the front end.	5	Web and Console menu Options to add/alter/delete a customer	5	JavaDoc and Testing including Unit Testing	10
		CRUD method to update book details based on ID	5						
Section Total	10		20		20		15		35

Your work in this assignment will be graded in a number of areas, attracting a number of marks for each. Guidance on the assessment criteria for each area is included below. The marks given for each area will be combined to give an overall mark for 1CWK100. For example, a student earning 10 marks in the *Product Class* section, 20 marks in the *Database Connectivity* section, 10 marks each in the *Customer* and *Web Server* sections and 5 marks for *JavaDoc* would achieve an overall mark of 55% (10 + 20 + 10 + 10 + 5). If you need any help in understanding this assignment please talk to the tutor who takes you for your laboratory sessions or arrange to see Dr Darren Dancey.

