

# INFO221 : Application Software Development

## Project Specification

### Introduction

The project that you will be working on in this course is an online shopping system.

In phase 1 you will be creating the administration part of the system that allows the shop staff to add new products to the system.

In phase 2 you will be creating the web based user interface that the customers will use to purchase products from the shop.

You will be working on the project throughout the entire course. Each week there will be a project task that is related to the topic covered in the lab that week. By completing each project task each week you will eventually end up with a complete project.

We **strongly** recommend that you keep on top of the project tasks and don't fall behind. This is not a trivial project, and you will not be able to leave it until the last week and complete it.

## 1 Phase 1

### 1.1 Requirements Analysis

The requirements analysis identified five use cases that need to be implemented in this phase of the project:

- A user needs to add a new product.
- A user needs to view all of the products.
- A user needs to view the products in a selected category.
- A user needs to edit the details of an existing product.
- A user needs to delete an existing product.

### 1.2 System Analysis

The analysis and design phase of the development has been completed and the following is some of the design documentation that describes the system.

## Graphical User Interface Mock-ups

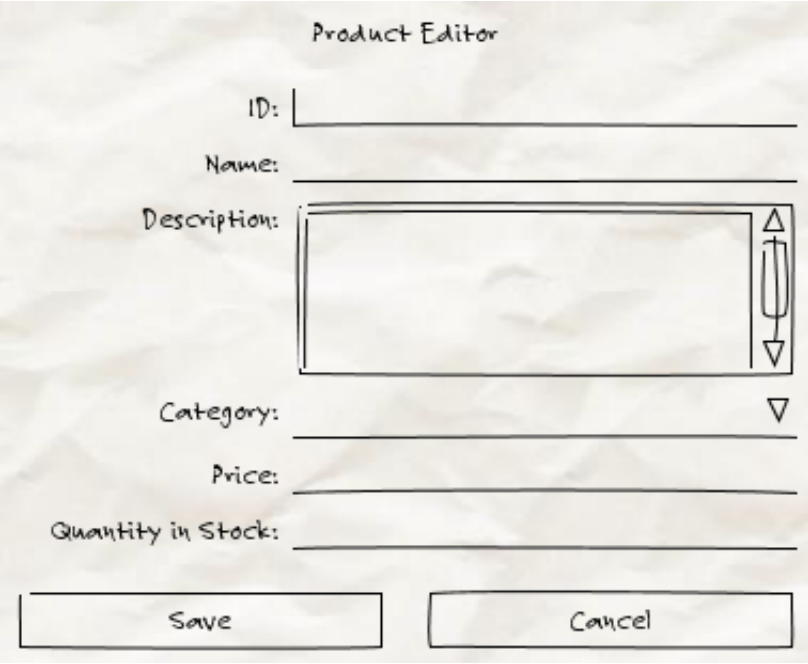
The following are low fidelity mock-ups of the forms for the GUI:



The Main Menu mock-up is a rectangular window titled "Flanges and Widgets 'R Us - Product Administration". It contains three large, horizontally-oriented buttons stacked vertically. The top button is labeled "Add a New Product", the middle button is labeled "View Products", and the bottom button is labeled "Exit".

The Main Menu

This could be a button based menu (as shown) or a standard pull down menu with the company's logo in the centre of the frame.



The product editor mock-up is a rectangular window titled "Product Editor". It contains several input fields: "ID:" with a single-line text box, "Name:" with a single-line text box, "Description:" with a multi-line text area, "Category:" with a drop-down list, "Price:" with a single-line text box, and "Quantity in Stock:" with a single-line text box. At the bottom of the window are two buttons: "Save" and "Cancel".

The product editor

This would be displayed when the user selects "Add a New Product" from the main menu.

The 'Category' field is an editable drop down list (JComboBox in Swing) that allows the user to either type in a new value or select an existing value from the drop down list.

A hand-drawn sketch of a software window titled "View Products". At the top, there is a label "Category: All" followed by a green downward-pointing triangle indicating a dropdown menu. Below this is a rectangular box containing three lines of text: "ID: FL1234, Name: Fuzzy Flange", "ID: FL7833, Name: Polkadot Flange", and "ID: WD8234, Name: Flimsy Widget". At the bottom of the window are three buttons labeled "Edit", "Delete", and "Close".

Viewing all products

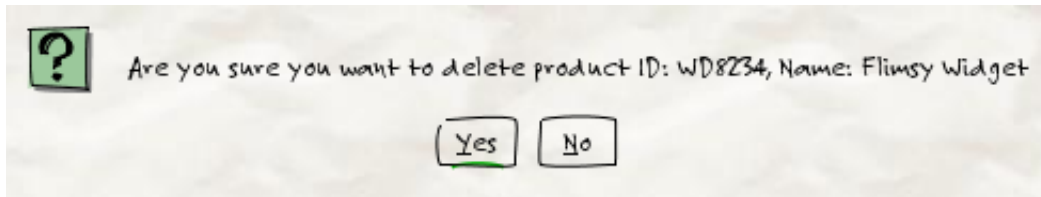
This would be displayed when the user selects “View Products” from the main menu.

By default the all of the products should be displayed. All of the categories that are in the system are contained in the drop down list.

A hand-drawn sketch of a software window titled "View Products". At the top, there is a label "Category: Flanges" followed by a green downward-pointing triangle indicating a dropdown menu. Below this is a rectangular box containing two lines of text: "ID: FL1234, Name: Fuzzy Flange" and "ID: FL7833, Name: Polkadot Flange". At the bottom of the window are three buttons labeled "Edit", "Delete", and "Close".

Viewing products in a specific category

If a user selects a category in the drop down list then only the products in that category should be displayed.



Deleting a product

This is shown when the user selects a product and clicks the “Delete” button.

Editing an existing product

This is shown when the user selects a product and clicks the “Edit” button. The selected product will be loaded into the product editor so that its details can be edited.

All user interfaces should be able to be resized properly. Text and list components should resize dynamically when the window they are contained within is resized.

**NOTE: The final implementation of this system doesn’t have to be identical to these low-fi mock-ups but should have similar functionality.**

## System Architecture

The system should be separated into four packages:

- The “domain” package that contains the classes relating to the object-oriented domain model of the business (see page 6 for details about the domain model).
- The “gui” package that contains the user interface classes for the staff product administration GUI.

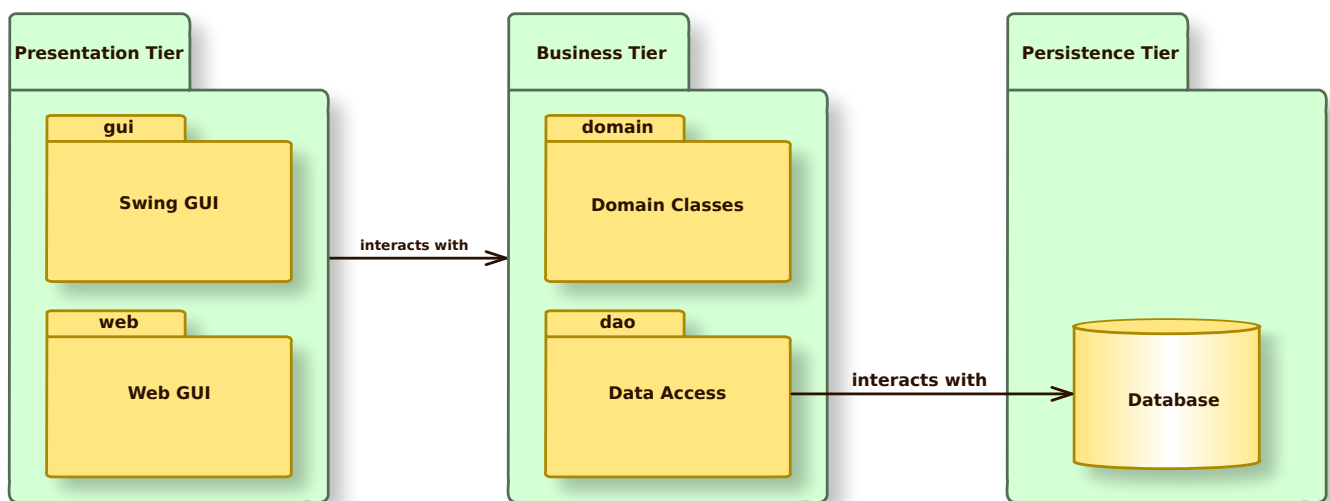
The user interface classes should be created using Swing.

- The “dao” (data access objects) package that contains the classes that are responsible for storing, finding, and managing the domain objects (products, customers, etc.) that have been entered into the system.

The data persistence will be implemented via JDBC with the data being stored in a relational database.

- The “web” package that will contain the classes associated with the web based user interface that will be created in the second phase of this project.

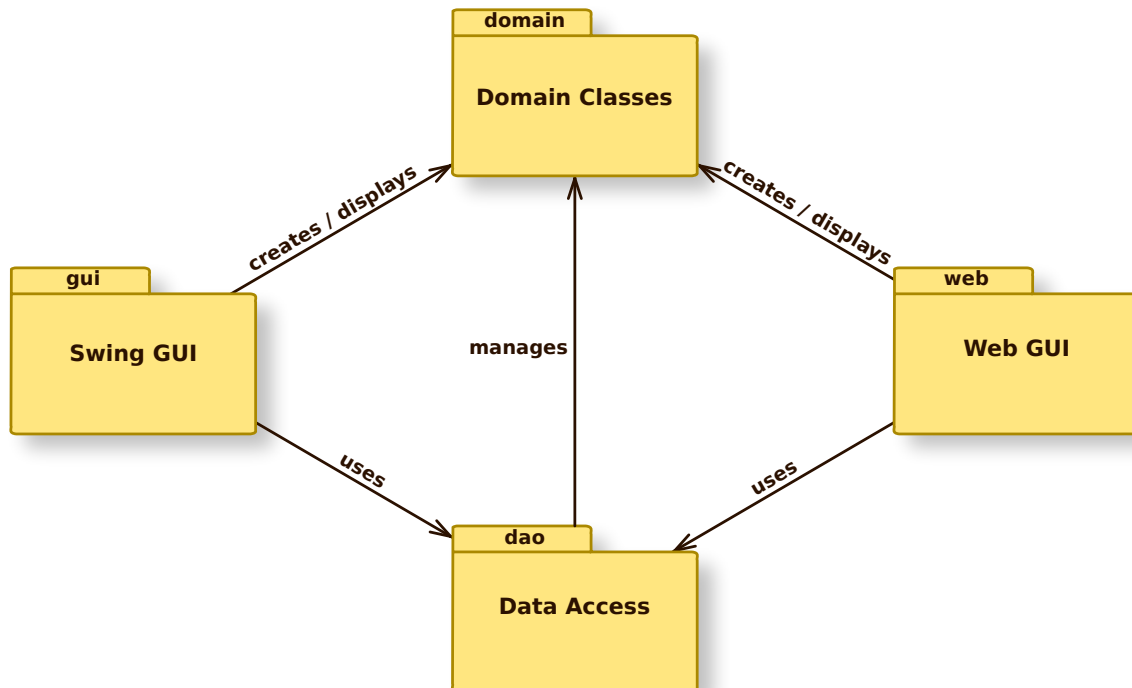
The architecture for this system is a logical three-tier system (as opposed to a physical three-tier system where each tier resides on a different computer and is connected by a network):



The three tiers

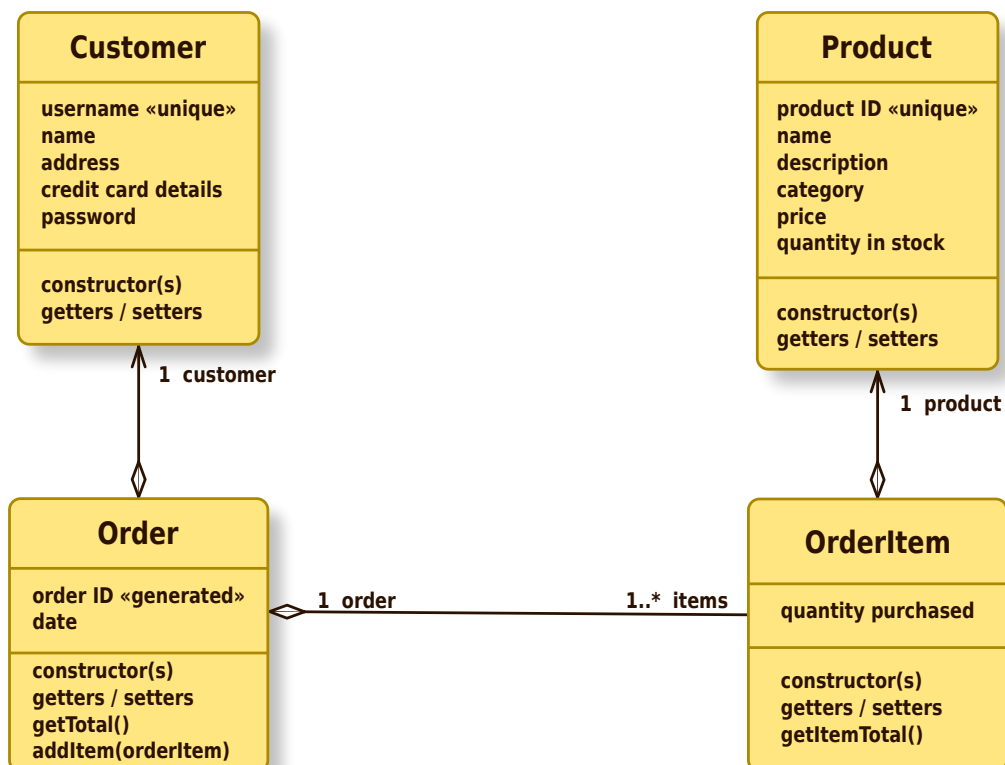
The presentation tier provides the user interfaces, the business tier provides the domain objects and the classes that manage them (the data access objects), and the persistence tier which in the case of this project is a database provided by a DBMS called H2 (<http://h2database.com/>).

Using this architecture gives us better control over the dependencies between the different packages:



The dependencies between the packages

## Object-Oriented Domain Model



Class Diagram of Domain Model

**Product** - Contains information about a product that is offered for sale at the shop.

**Customer** - Contains the account details for a customer of the shop.

**Order** - Contains information about a single transaction made at the shop.

**OrderItem** - Represents how much of a single product was purchased in an order.

## 1.3 Unit Testing

Ideally the entire system should be unit tested as it is developed, but due to time constraints we will only write unit tests that ensure the main use cases are implemented correctly.

All testing should be done using JUnit and FEST inside of NetBeans as shown in lab 6.

You are required to test the CRUD (create, read, update, delete) operations for product as follows.

**Create/Add** – Write testing code that enters data into the product dialog and clicks the save button. The DAO should now have an additional object in it, and that object should contain the details that were entered into the GUI.

**Read/View** – Write testing code that opens the product report, and counts the number of objects being displayed. The count should be the same as the number of objects in the DAO. The objects that are in the report should be the same objects that are in the DAO. You should also test that the text being displayed in the list (the output of the product object's `toString`) contains the details that you want to be displayed.

**Update/Edit** – Write testing code that selects an object on the report and clicks the edit button. The testing code should then check the contents of the edit dialog that was opened to ensure it is displaying the correct details. The testing code should then change some of those details and click the save button. The DAO should have no change in the number of objects that it stores, but the object that was edited should contain the new details.

**Delete** – Write testing code that selects an object on the report and clicks the delete button. The DAO should now have one less object in it, and should no longer contain the object that was deleted.

You should make sure that any objects that your tests have added are also deleted again to prevent them from causing database constraint violations when other tests run. You can either do this at the end of each test, or use the `@After cleanup` method.

## 1.4 Phase 1 Requirements

For this first phase of the system you are required to:

1. Complete all of the domain classes shown on page 6.
2. Create a JDBC based DAO class for the product domain object.

3. Create GUIs that allow the user to perform the five use cases described in section 1.1 on page 1.
4. Use JDBC for storing product objects in a database and for performing all persistence related (DAO) operations.
5. The system should gracefully handle any input errors made by the user. This means rather than crashing and displaying a stack trace in the console the system should catch any exceptions caused by user input error and display a relevant explanation to the user in a message box.
6. Create unit tests for testing the system as described in section 1.3 on page 7.

### **Minimum Requirements for Project Submission**

As per the course outline we enforce a minimum acceptable standard for project submissions.

In order to get any marks at all for this phase your system must at have at least the following functionality:

- The system must allow the user to create a product via the user interface.
- The details of that product must be stored in a database using JDBC via a DAO.



## 2 Phase 2

In this phase you are adding a dynamic web site to the system that will allow customers to browse and purchase the products that have been entered into the database.

You will be using a combination of Java servlets and JSP to do this.

### 2.1 Use cases for the web site:

- A new customer should be able to create a new account. The customer's details (as shown in the class diagram on page 6) should be stored in the database.
- An existing customer (one who has already created an account) needs to be able to log in using the usercode and password that they entered when they created their account.
- The customer needs to be able to view all of the products that have been stored in the database, and select one for viewing.
- Once the customer has selected a product all of its details should be displayed, and the customer should be able to enter the quantity of this product that they want to purchase.

An order item object should be used to hold the details of the product and the quantity. See the description of the order item class on page 6.

The order item should be stored in a collection in the session.

- The customer needs to be able to 'check out' their order.  
This will cause an order object to be created. All order item objects stored in the session should be moved into the order object and stored in the database along with the order. The quantity in stock values of the products that were purchased should be decremented by the amount that the customer purchased.  
The order items in the session should be cleared out once the transaction is complete in case the user wants to carry on shopping.
- The customer needs to be able to log out. This should clear any session data that was associated with this customer.
- When logged in the user should always be able to see navigation links that will take them to the products page, the check out page, or logout.

### 2.2 Phase 2 Requirements

The system needs to provide a web site that provides all of the functionality required to implement the use cases mentioned in the previous section.

You will be given more information about how to implement this functionality during labs and lectures.

## **Minimum Requirements for Project Submission**

In order to get any marks at all for this phase your system must at have at least the following functionality:

- A customer must be able to create an account via a web page using JSP or servlets.
- The account details entered must be stored in a database using JDBC, JSP or servlets.
- The customer must be able to log into the system using their usercode and password via a web page using JSP or servlets. If the usercode and password are not correct then they should not be allowed to log into the system.
- The user must be able to view a web page that dynamically displays a list of all of the products that are stored in the database using JDBC, JSP and servlets.