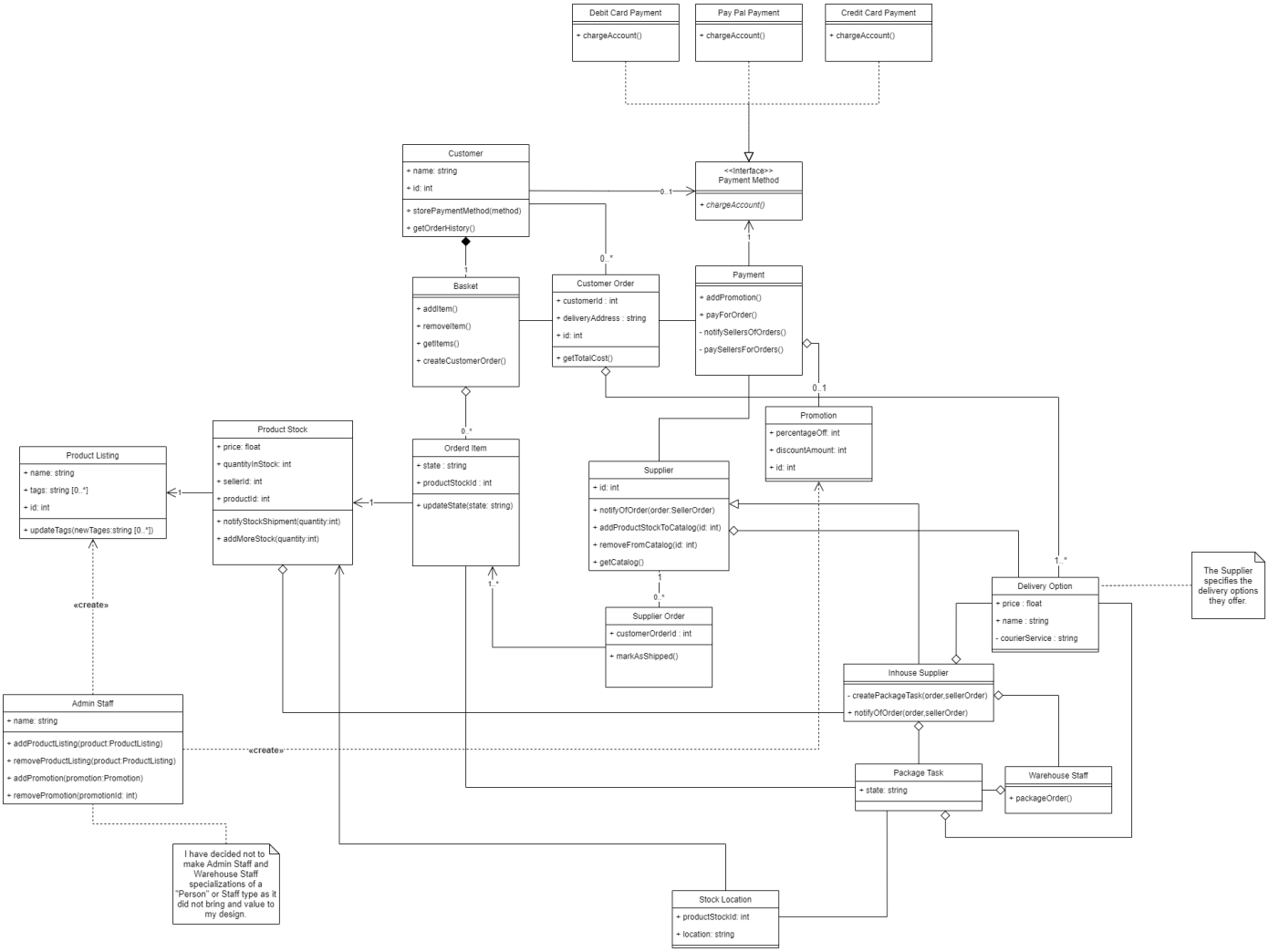
E-Store System Design

This document features my object orientated modeling for an online E-store.

I have modeled, using UML, static and dynamic visualizations of the system to describe the key objects and their interactions.

Class Diagram



During the development of this diagram, I identified the key objects in the system and assigned responsibilities to them.

Overview

Central to the functioning of any store are the products available for purchase. I have identified two objects that describe the products in my system design; **Product Listings** and P**roduct Stocks.**

The **Product Listing** object represents a product that can be sold through the store whilst a **Product Stock** represents the physical supply of product. A **Product Listing** on the store may be available for

purchase from multiple sellers, therefore I modeled a zero to many relationship for **Product Listings** to **Product Stocks**.

Adding a **Product Listing** means the store has the ability to sell that product. I considered two likely creators of the **Product Listings**, either the sellers or the store administrators. I realized a problem with **Sellers** having this responsibility is that it could lead to undesirable or illegal products being listed on the site. As such I think it best for Admin staff to be responsible for deciding which products and promotions are available on the site.

Orders

There are three objects in the system that are used to represent orders, these are the **Customer Order**, **Supplier Order** and the **Ordered Item**.

An **Ordered Item** represents an item that a **Customer** is ordering, this object has a reference to the item that has been ordered as well as the state of that order i.e.(awaiting picking, payment pending).

A **Customer Order** encompasses the whole order a **Customer** has made. One **Customer Order** may feature **Ordered Items** from multiple sellers.

A **Seller Order** is the object that describes to a **Seller** which products have been ordered.

Payment

The **Payment** object handles the transaction of paying for a **Customer Order**. The **Payment** object requires a **Payment Method** and a delivery address.

The **Payment** object can also optionally accept a **Promotion**. Once the payment is complete the **Sellers** are notified of the orders.

Sellers

Sellers are able to add Products Stock. Without **Product Stock** it is not possible for **Customer** to purchase a product. When **Sellers** add a **Product Stock**, the **Product** is then visible within the **Sellers** catalog.

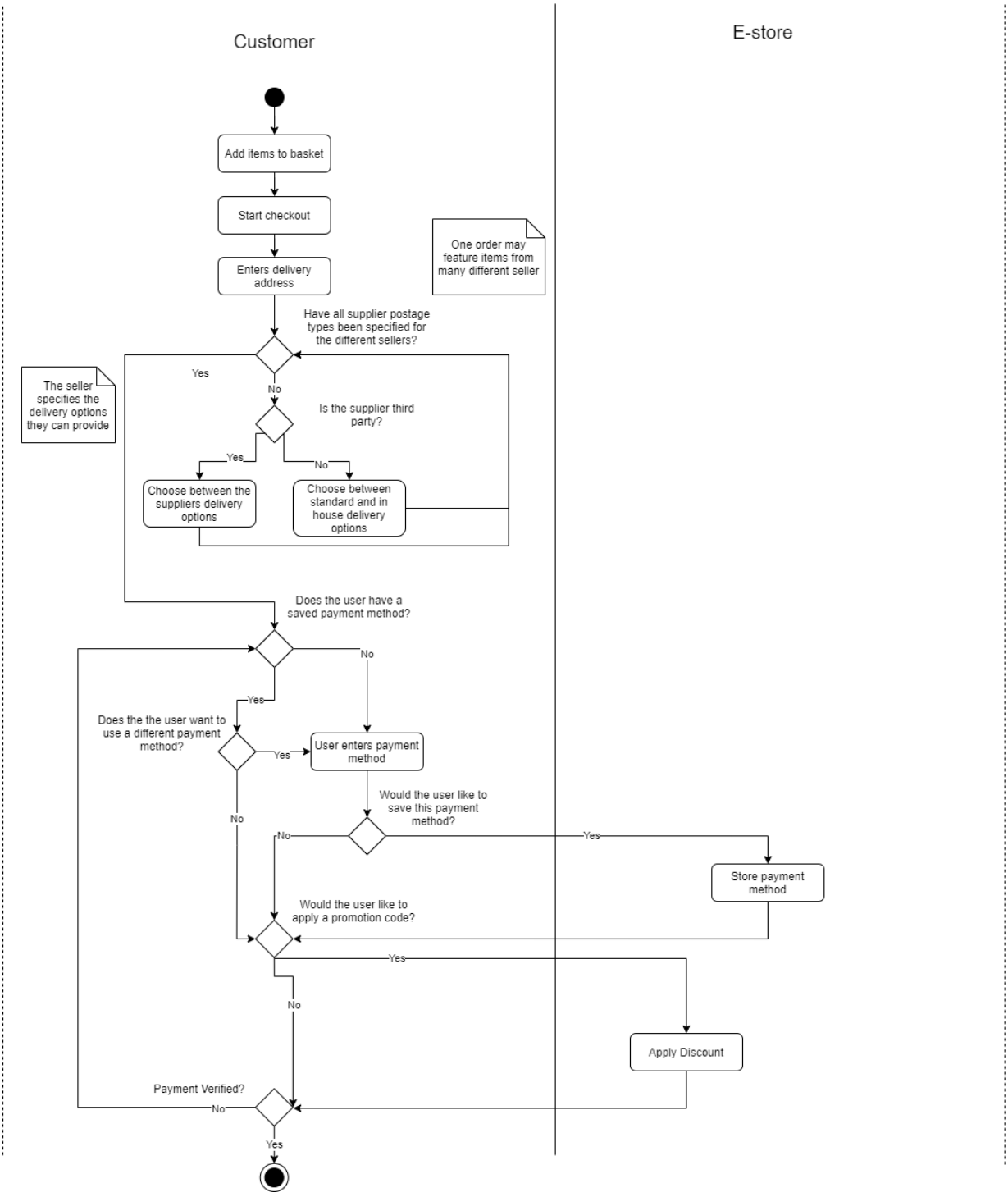
Inhouse Seller

The e-store website itself will stock and sell some products, I have modeled the website as special type of **Seller**.

The **Inhouse Seller** is a specialized **Seller** that inherits from the **Seller** object. The **Inhouse Seller** has additional responsibilities for triggering the packaging and collection of products from the warehouse.

A **Package Task** is an object created by the **Inhouse Seller** that has a reference to the **Ordered Items** that need to be packaged. The **Stock Location** object maps **Product Stocks** to locations that **Warehouse Staff** can use to pick **Seller Orders**.

Activity Diagram – A Customers Order Completion Process



This activity diagram shows the procedural logic for a **Customer's** order.

The **Customer** object represents a user who wants to purchase from the site. Every **Customer** has a basket. When **Customers** decide to purchase products, an **Ordered Item** is added to their basket.

State Diagram – Order States