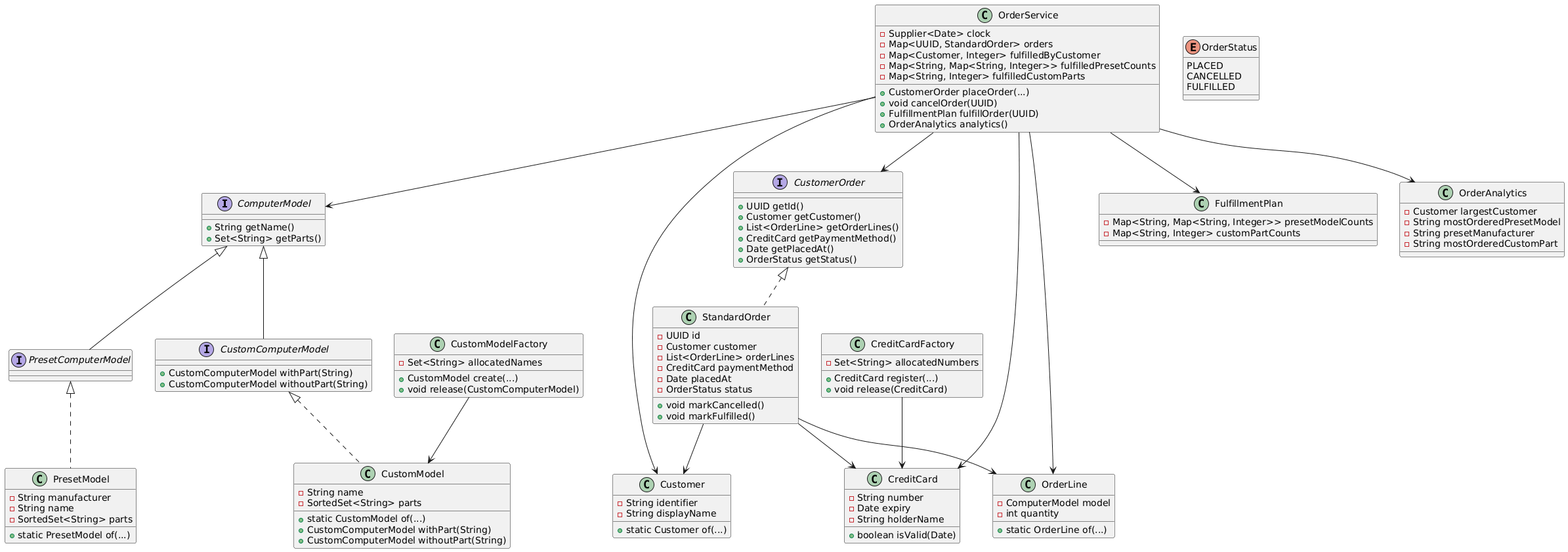
**STUDENT NO :** 250574858

**NAME:** khwaja nawaz s

**CSC8404 - PC Retailer Order System – Design Overview**

# UML Class Diagram



**Figure:** Core interfaces and classes with their key relationships.

**This report is a summary of the PC retailer ordering system, its key packages, interfaces, and classes.**

**Overall Structure**

* Domain model package: This package defines preset and custom computer models are the immutable value objects.
* Customer and payment packages offer verified documents of buyers and their credit cards.
* Place order packages, cancel, fulfil and analytics using a service facade.

# Key Interfaces

* **ComputerModel:** Base contract exposing model identity and the set of parts.
* **PresetComputerModel:** Extends ComputerModel to include the originating manufacturer.
* **CustomComputerModel:** Adds immutable part addition and removal operations that return new instances.
* **CustomerOrder:** Read-only view on orders so clients cannot bypass lifecycle checks.

# Principal Classes

* **PresetModel:** Immutable preset machine; validates manufacturer, name, and parts.

it saves everything in an immutable way. That keeps things safe. It includes equals and hashCode methods. ToString gets added as well.

* **CustomModel:** Immutable custom machine supporting copy-on-write part changes.

Immutability keeps everything secure. Predictability stays intact that way. Validation covers every input detail. Names and parts get checked thoroughly. It includes equals, hashCode, toString so you can compare, store in sets and maps, and print models easily.

* **CustomModelFactory:** Tracks allocated custom names to enforce uniqueness.

It creates custom computer models while making sure each model name is unique no duplicates, case-insensitive. It validates inputs and keeps track of used names, with a way to release a name again mainly for tests.

* **Customer:** Immutable identifier/display-name pair with non-blank validation.

This code is used to create a customer with a unique Id and name in a safe. It checks that both Id and name are not empty, and makes sure each customer is identified only by their Id.

* **CreditCard:** Immutable credit card record providing defensive copies and expiry checks. This code safely represents a credit card with its number, expiry date, and cardholder’s name that cannot be changed once created.

* **CreditCardFactory:** Allocates unique card numbers and constructs CreditCard objects. The code handles creation of credit cards. It makes certain that every card gets a unique number, no duplicates at all. Finally, each card ends up secure in its details. No one can change them once they are set.
* **OrderLine:** Pairs a ComputerModel with a positive quantity for basket lines. This code stands for a single item from what a customer orders. It connects a particular computer model to how many were purchased, the setup checks that the model actually exists. It also confirms the quantity comes out greater than zero.
* **StandardOrder:** Concrete CustomerOrder controlling status transitions internally.

This code sets up an order object designed to stay fixed once created.The initial status gets set to PLACED right from the start. Changes to the order status happen only through specific paths. Those lead to either CANCELLED or FULFILLED. Copies of certain elements, such as dates and lists, get created separately. That setup helps ensure no external factors can alter the original by mistake.

* **FulfillmentPlan:** Collates manufacturer model totals and part counts for fulfilment. This class manages the pick list used in the warehouse. It tracks the count of preset models, which get grouped by manufacturer and then by model. The class creates protected copies of the maps that cannot be changed.
* **OrderAnalytics:** Stores cumulative statistics such as top customers and parts. This class holds onto basic top statistics pulled from completed orders. Those stats cover the biggest customer by volume, and the top custom part that buyers picked most often. Everything is read-only and returned as Optional, so callers can safely check if a stat exists without getting errors.
* **OrderService:** Central façade coordinating placement, cancellation, fulfilment, and analytics updates. This class runs the whole order flow: place an order, cancel it, or fulfill it, while checking inputs and card expiry. When fulfilling, it builds a warehouse plan preset models and custom parts and locks the order as fulfilled.

# Order Lifecycle and Analytics

The OrderService checks card details, currency, and order lines before any placement happens. Cancellation can occur right up to the point of fulfillment. Fulfillment then generates a FulfillmentPlan. This plan pulls together requirements set earlier by the manufacturer, along with parts for custom builds. The operation also syncs cumulative analytics at the same time. That means no need to replay history just to query top customers, preset models, or custom parts.