ANEXO EXAMEN DE CERTIFICACIÓN

Plan de Estudio	Desarrollo Aplicaciones Fullstack Java Trainee
Anexo	Caso Classicmodels Inventory & Orders

Caso “Classicmodels Inventory & Orders”

“Classicmodels” es una empresa dedicada a la fabricación y comercialización de modelos a escala de distintos tipos de vehículos motorizados tales como automóviles, motocicletas, trenes y aviones, entre otros. Sus modelos son de alto realismo y de gran calidad, razón por la cual ha logrado posicionarse en el mercado de los coleccionistas a nivel mundial.

La empresa se creó en los años sesenta como una empresa familiar, y debido a la buena calidad de sus modelos, fue rápidamente creciendo. Actualmente, cuenta con oficinas en las principales capitales del mundo, tales como New York, Boston, San Francisco, Paris, Tokyo, Sydeny, y London.

El crecimiento de la empresa no ha estado exento de dificultades, de hecho, el principal reclamo es el atraso en los pedidos debido a que no se lleva un buen control de las existencias y las órdenes. Al respecto, el CEO de la organización ha solicitado al CTO que impulse un proyecto de mejora de los sistemas de inventarios y existencias que ya data de los años 90s y no se le han hecho grandes mejoras.

El CTO está muy entusiasmado con el proyecto y rápidamente ha formado un equipo de proyectos de primera línea del cual usted forma parte como desarrollador full-stack Java. El equipo de proyectos también lo conforma un Jefe de Proyectos, un Diseñador UX/UI, un Diseñador Web, un Analista Funcional, un Desarrollador Mobile, y un Arquitecto de Software.

El proyecto busca, dentro de otras cosas, ordenar el sistema de inventario y de órdenes que se cursan desde las distintas oficinas. A continuación, se listan los requisitos funcionales de alto nivel del sistema:

- El sistema debe permitir la consulta de inventarios y existencia de los productos de las distintas líneas (automóviles, motocicletas, trenes, aviones, etc)
- El sistema debe permitir la consulta de las órdenes que se encuentran en proceso
- El sistema debe permitir el cálculo de descuentos y promociones al momento de ingresar una orden

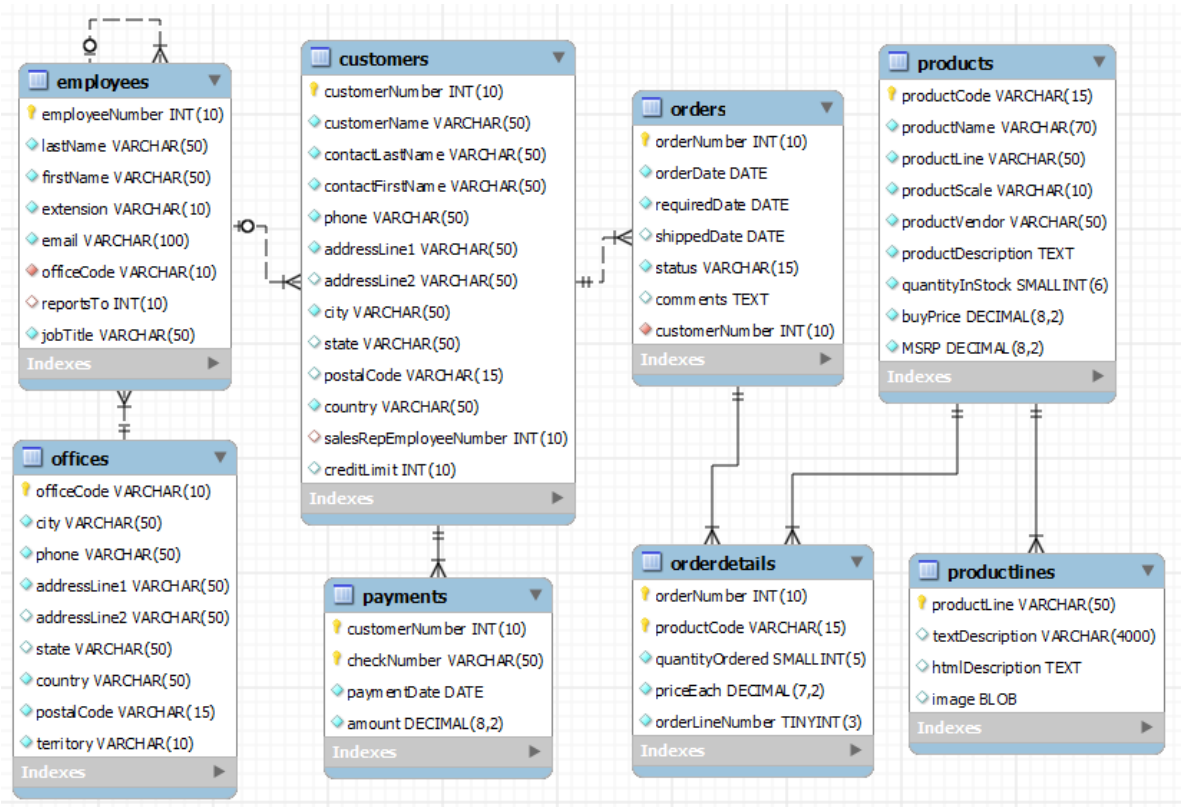
A la fecha, ha transcurrido gran parte del proyecto y se tiene el siguiente avance:

- Ya se cuenta con un prototipo funcional del aplicativo
- Existe un modelo de datos diseñado

- Existe una base de datos con datos de prueba
- Existe una aplicación web desarrollada con Spring Framework que desarrolla algunas de las funcionalidades requeridas

Modelo de Datos

A continuación, se presenta el modelo de datos diseñado por el arquitecto en conjunto con el analista funcional.



Como se puede observar, el modelo contiene las órdenes de la compañía en una relación master-detail entre las tablas *orders* y *ordedetails*.

Por otra parte, existe un maestro de productos (tabla *products*) que mantiene los principales atributos de cada producto, en donde se encuentra el código de producto, nombre, datos de stock (quantityInStock) y precio de costo (buyPrice), entre otros.

Los productos están agrupados en líneas de producto, para así poder administrarlos de mejor manera. La tabla *productlines* entrega la información de cada línea de productos, el cual es utilizado por los category managers para hacer gestión por categoría.

El maestro de oficinas (*offices*) contiene información de cada sucursal. Cada sucursal, a su vez, maneja un grupo de vendedores y managers los cuales se encuentran almacenados en la tabla *employees*. Cada vendedor maneja su propia cartera de clientes, lo cual se ve reflejado en la relación con la tabla *customers*.

La tabla *employees*, no solamente alberga a los vendedores y managers sino toda la estructura de la organización, con todos los miembros de la compañía en una relación jerárquica a través del campo *reportsTo*.

El maestro de clientes (tabla *customers*) contiene la información general de cada uno de los clientes de la compañía y a su vez se relaciona con la tabla de órdenes (*orders*) y de pagos (*payments*).

Requerimientos a Desarrollar

El jefe de proyectos, quien lleva un control meticuloso de las actividades del proyecto, le ha solicitado a Usted que realice las siguientes tareas:

1. Realizar consultas a la base de datos
2. Construir un algoritmo de cálculo de promociones y descuentos
3. Construir una unidad de pruebas para verificar el correcto cálculo de las promociones y descuentos
4. Crear Monitor de Ordenes
5. Crear una API REST que disponibilice la información del monitor de ordenes

A continuación, se especifica con mayor detalle cada uno de los requerimientos:

1. Realizar consultas a la base de datos

El CTO solicitó realizar algunas consultas a la base de datos para la extracción de cierta información necesaria para el negocio, mientras se termina el desarrollo de la aplicación. **Para esto, cree un package en su proyecto Java con nombre “consultas”, cree un archivo por cada una de ellas, identificando claramente de qué consulta se trata** (ejm: Consulta-A.sql, Consulta-B.sql, ...).

a) El category manager de Motorcycles necesita saber cuáles son los productos más inventariados, para hacer una campaña de ventas con esos productos. Se necesita crear una query que retorne el listado de productos de la categoría “Motorcycles”, con su inventario ordenado de mayor a menor. Se requiere obtener un listado de la siguiente forma:

productCode	productLine	productName	quantityInStock ▼ 1
S12_2823	Motorcycles	2002 Suzuki XREO	9997
S32_2206	Motorcycles	1982 Ducati 996 R	9241
S10_1678	Motorcycles	1969 Harley Davidson Ultimate Chopper	7933
S18_3782	Motorcycles	1957 Vespa GS150	7689

b) El Gerente Comercial desea saber cuántos productos en stock se tiene de la marca "*Ford*" en las distintas líneas de producto, ordenado alfabéticamente por línea de producto. El reporte debe tener la siguiente forma:

productLine ▲ 1	productCode	productName	quantityInStock
Classic Cars	S18_4933	1957 Ford Thunderbird	3209
Classic Cars	S12_1099	1968 Ford Mustang	68
Classic Cars	S12_3891	1969 Ford Falcon	1049
Classic Cars	S18_3482	1976 Ford Gran Torino	9127
Trucks and Buses	S18_2432	1926 Ford Fire Engine	2018

c) El Area Manager desea saber cuántos productos tiene cada línea de producto, el resultado debe estar ordenado de mayor a menor número. El reporte debe tener la siguiente forma:

productLine	Cantidad
Classic Cars	38
Vintage Cars	24
Motorcycles	13
Planes	12
Trucks and Buses	11
Ships	9
Trains	3

d) El departamento de logística y despacho desea un listado con las órdenes en estado "In Process" que hay en este momento en el sistema

y que hayan solicitado el producto con código “S18_179”. El listado debe lucir de la siguiente forma:

orderNumber	orderDate	quantityOrdered	priceEach
10420	2005-05-29	37	153.00

e) El Presidente de la compañía desea un listado con el ranking de venta de productos por línea durante el año 2004. El reporte debe lucir de la siguiente manera:

productLine	sum(odt.quantityOrdered)
Classic Cars	16085
Vintage Cars	10864
Motorcycles	5976
Planes	5820
Trucks and Buses	5024
Ships	4309
Trains	1409

2. Construir un algoritmo de cálculo de promociones y descuentos

Un objetivo importante de este proyecto es desarrollar un algoritmo avanzado para el cálculo de las promociones y descuentos que se aplicarán al momento de cursar una orden en el sistema. De esta forma, los vendedores podrán otorgarles descuentos más atractivos a sus clientes y así impulsar la venta.

Para lograr lo anterior, se le pide que desarrolle un algoritmo de cálculo de descuentos y promociones con distintas lógicas de cálculo. Posteriormente, en la medida que se aprecien buenos resultados, se irán extendiendo e incorporando nuevas lógicas y reglas de negocio.

Construya dos algoritmos de cálculo de promociones, uno que realice **cálculos simples** y otro que realice **cálculos complejos**.

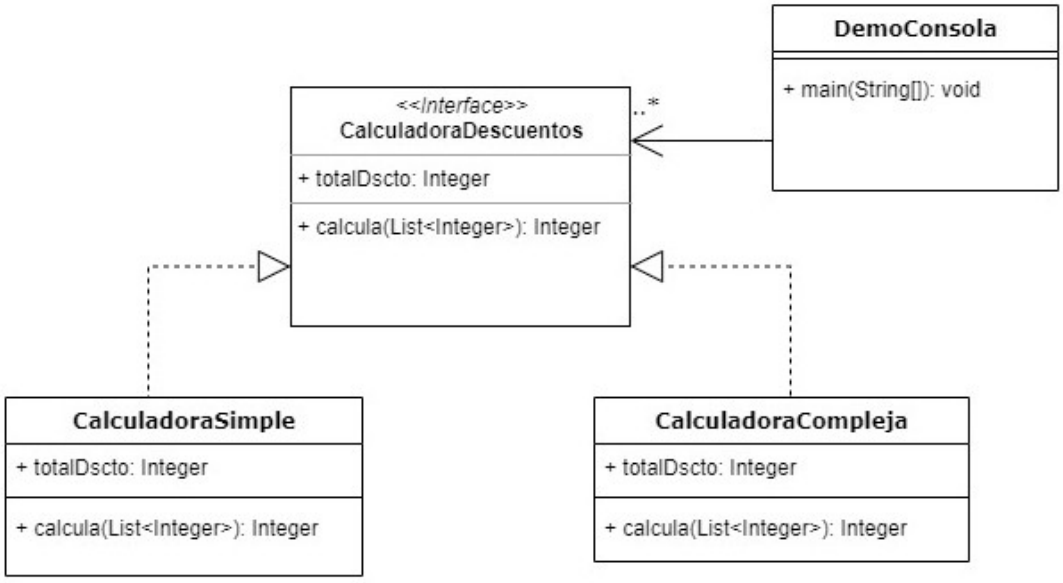
Las reglas del algoritmo de **cálculo simple** son las siguientes:

- El algoritmo debe aplicar un 10% de descuento en el precio a todos los productos (se asume que cada precio es de 1 producto unitario)

Las reglas del algoritmo de **cálculo complejo** son las siguientes:

- El algoritmo debe aplicar un 10% de descuento a los precios que superen los $5.000. Pero si la suma de todos los precios supera los $20.000, debe aplicarse un 15% de descuento.
- Aplicar un 5% de descuento a los productos cuyo precio es igual o inferior a $3.500

Ambos algoritmos deben recibir un listado de precios enteros y retornar el cálculo del monto total de descuento redondeado (valor entero). El arquitecto del proyecto le sugiere que realice un diseño de clases similar al del siguiente diagrama:



Para hacer una demostración de los algoritmos, cree una **aplicación de consola** que genere 5 precios aleatorios entre $1.000 y $10.000 y que posteriormente realice el cálculo de los descuentos utilizando las dos implementaciones creadas anteriormente (algoritmo simple y complejo). Al ejecutarse, debería funcionar de la siguiente forma:

```
--------------------------------------
Demostración Calculadora de Descuentos
--------------------------------------

Tomando 5 precios aleatorios...
2432 6523 8172 3341 9832

Descuento con Algoritmo Simple: 3030
Descuento con Algoritmo Complejo: 3968
```

Considere hacer un diseño de clases polimórfico, mediante interfaces, en donde deberá crear una interfaz y dos clases concretas que implementen dicha interfaz. Una de ellas que implemente el **algoritmo de cálculo simple** y otra que implemente el **algoritmo de cálculo complejo**. Asimismo, una clase que sea la que ejecuta la aplicación de consola y genera los números aleatorios para entregárselo a cada uno de los algoritmos.

Genere dentro de su proyecto en eclipse un package con nombre representativo que tenga las clases mencionadas.

3. Construir una unidad de pruebas para verificar el correcto cálculo de los descuentos

Construya una clase de pruebas en Java que permita verificar el correcto funcionamiento de la clase que realiza el **cálculo complejo de descuentos**, considerando al menos los siguientes tests:

- Tests que considere casos normales, con distinta cantidad y valores de precio.
- Tests que considere condiciones de borde, por ejemplo, qué pasa cuando viene un precio cero, u otras condiciones de excepción o de borde.

4. Crear Monitor de Ordenes

Se requiere crear una página web dinámica que despliegue el listado de órdenes, tal como se detalla en la siguiente imagen mock-up.

Se pide:

a) En el combobox de **Estado**, desplegar un listado con valores fijos. Los posibles estados de una orden son los siguientes: Shipped, Cancelled, Resolved, On Hold, In Process, Disputed.
b) En el combobox de **Cliente**, desplegar el listado de nombres de clientes a partir de la información almacenada en la base de datos, ordenado alfabéticamente.
c) En los campos **Fecha Orden Desde** y **Fecha Orden Hasta**, debe permitir el ingreso de una fecha validando el formato para que sea el correcto (puede utilizar un validador o bien un elemento de tipo calendario, lo que prefiera).
d) Al presionar el botón **Buscar**, deberá desplegar el lisado de órdenes que cumplen con filtros seleccionados. Todos los filtros son opcionales para realizar una búsqueda.

Para realizar el requerimiento, el arquitecto le señala lo siguiente:

- Utilizar la tecnología de vista de su predilección (jsp, taglibs, thymeleaf)
- Utilizar los elementos del framework bootstrap
- Verificar que su interfaz se ajusta a distintos tamaños de pantalla

5. Crear una API REST que disponibilice el listado de ordenes

El CTO está pensando en crear una aplicación mobile para los vendedores, razón por la cual necesita que se cree una API REST para la consulta de órdenes, similar al módulo solicitado del punto anterior.

Para esto, se le solicita la creación de un servicio REST que permita obtener la misma información del monitor de órdenes, pero acorde a la especificación REST. Recuerde que el servicio podría recibir los mismos parámetros de búsqueda del punto anterior (estado, cliente, fecha de la orden desde y hasta).