ANEXO EXAMEN DE CERTIFICACIÓN

Plan de Estudio	Desarrollo Aplicaciones Fullstack Java Trainee
Anexo	Caso World-Parts Tech Catalog & Inventory

Caso “World-Parts Tech Catalog & Inventory”

La compañía “World-Parts Tech” es una organización dedicada a la distribución y comercialización de partes e insumos computacionales. Es un mayorista que compra en grandes volúmenes a las principales marcas de tecnología tales como Intel, AMD, Samsung, Western Digital, entre otros, y se encarga de distribuirlas en comercios minoristas, tiendas al detalle y servicios técnicos del rubro.

World Parts Tech es una compañía con fuerte presencia de mercado en países tales como Estados Unidos, Canadá, Italia y Japón. Para eso, cuenta con una fuerza de venta en terreno especializada y varios almacenes estratégicamente ubicados para la distribución de productos.

Una de las principales fortalezas de la empresa es el manejo de amplia variedad de productos y alta disponibilidad de inventario en sus almacenes para responder de forma rápida a las necesidades de aprovisionamiento de sus clientes, con costos competitivos debido a las economías de escala que posee.

Uno de los desafíos que World-Parts Tech se ha planteado para este año, es la renovación de su sistema de inventario y catálogo de productos que los vendedores en terreno utilizan para generar órdenes en conjunto con los clientes que atienden. El motivo de esto, es que el actual sistema fue construido con tecnología de hace 30 años atrás, razón por la cual su mantenimiento es dificultoso, asimismo su usabilidad.

Para esto, el CTO (Chief Tecnological Officer) 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 catálogo de productos que utilizan los vendedores en terreno para la gestión de los pedidos y ventas. A continuación, se listan los requisitos funcionales de alto nivel del sistema:

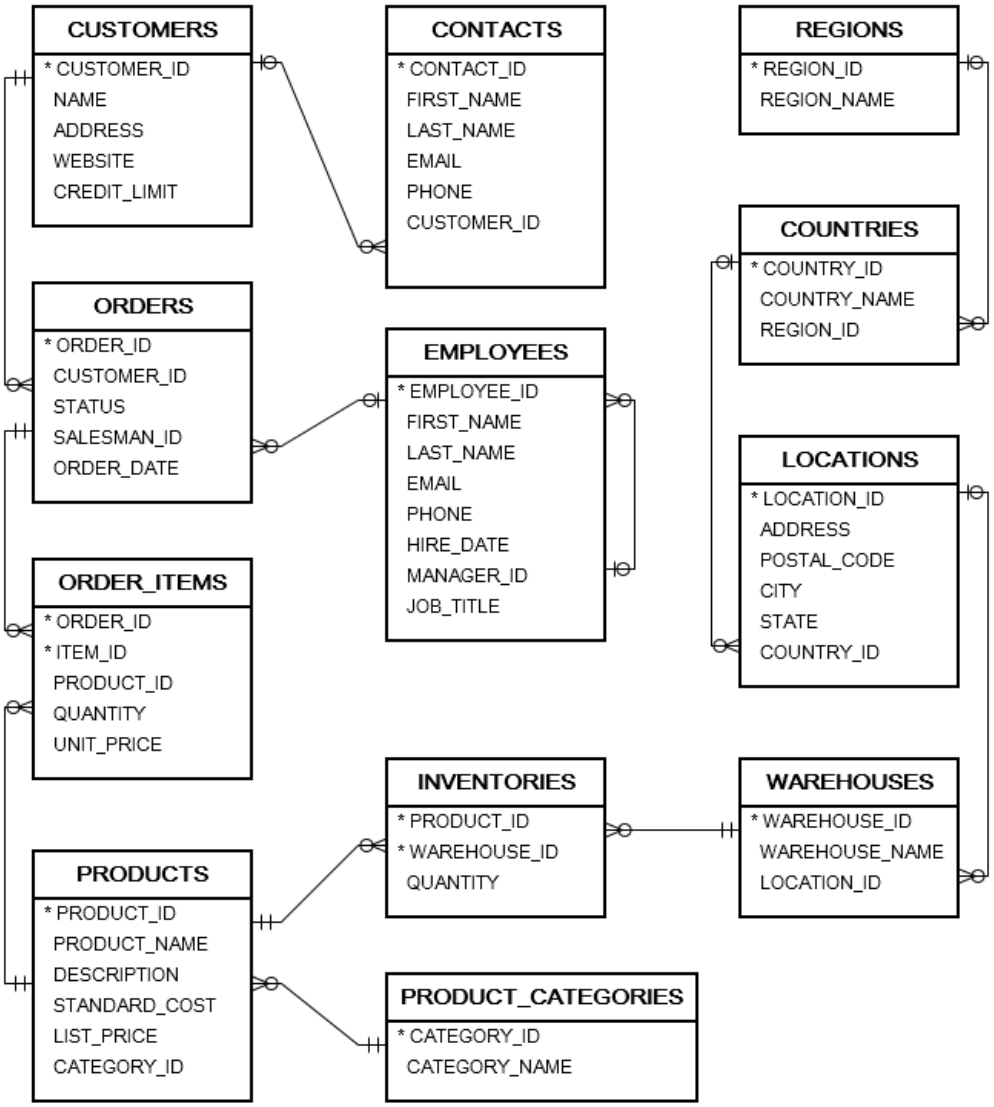
- El sistema debe permitir visualizar el catálogo de productos
- El sistema debe permitir visualizar el inventario de productos por almacén
- El sistema debe permitir la validación de los márgenes de venta

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.



La tabla *warehouses* corresponde a los almacenes que la compañía posee y que acumulan inventario de productos para ser comercializados y despachados. La tabla *locations* contiene todas las direcciones que la compañía posee, ya sea de almacenes, oficinas de venta y oficinas administrativas.

Cada almacén está conectado con alguna locación, con lo cual se puede determinar la dirección. Las locaciones, a su vez, están asociadas a los distintos países, los cuales están almacenados en la tabla *countries*. Por otra parte, como se trata de una compañía con presencia internacional, se cuenta con una catalogación de países en regiones del mundo (tabla *regions*).

La tabla *products* corresponde al maestro de productos de la compañía, en donde se almacenan atributos importantes del producto tales como el precio de lista y el costo estándar, a parte de su nombre y descripción. Los productos están relacionados con distintas categorías, las cuales están almacenadas en la tabla *product_categories*.

El inventario de los productos se encuentra en la tabla *inventories*, y almacena la cantidad en existencia de cada producto para cada uno de los almacenes. De ahí la relación tanto con la tabla products como con la tabla warehouses.

Por otra parte, el modelo cuenta con información de las órdenes que se van cursando (tabla *orders*), la cual tiene una relación maestro-detalle con la tabla *order_items*, que almacena los ítems que fueron pedidos en cada orden. La tabla *orders* maneja información general, por ejemplo, la fecha en que fue solicitada la orden, el estado en que se encuentra, el cliente y el vendedor, entre otra información. La tabla *order_items*, almacena la información de los productos asociados a cada orden, junto con la cantidad y precio unitario de éstos al momento de cursar la orden.

La tabla *employee*, almacena la información de los empleados, los cuales cumplen distintos roles dentro de la organización. Esta entidad tiene una relación consigo misma para representar las relaciones jerárquicas, es decir, empleado-jefe. Esta tabla está relacionada con las órdenes, con lo cual se puede saber qué vendedor realizó la venta de cada orden.

Por otro lado, la tabla *customers* corresponde al maestro de cliente, y contiene la información básica de cada empresa junto con el límite de crédito que tiene asignado para realizar compras. A su vez, la tabla customers está relacionada con la tabla *contacts*, que tiene información de los contactos válidos en cada empresa para realizar órdenes.

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 validación del margen de venta de productos
3. Construir una unidad de pruebas para verificar el algoritmo de validación del margen de venta
4. Crear un Monitor de Productos
5. Crear una API REST que disponibilice la información del Monitor de Productos

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

1. Realizar consultas a la base de datos

El Chief Sales Manager ha solicitado algunos reportes de información, razón por la cual el jefe del proyecto le ha encargado a usted que realice algunas consultas en la base de datos, para la extracción de cierta información necesaria para el negocio, por 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, …etc).

a) El Product Manager, solicitó un listado de todos los productos de la categoría 'Storage', con su nombre y precio, ordenados alfabéticamente por nombre. El reporte debe tener la siguiente forma:

product_id	product_name ▲ 1	list_price
33	ADATA ASU800SS-128GT-C	52.65
230	ADATA ASU800SS-512GT-C	136.69
79	Corsair Dominator Platinum	659.99
262	Corsair Dominator Platinum	1449.99
118	Corsair Dominator Platinum	699.89
37	Corsair Dominator Platinum	1264.99
35	Corsair Dominator Platinum	1314.99

b) El Product Manager, para realizar una gestión por categoría, requiere saber cuántos productos hay en cada categoría. Se le solicita un reporte con la cantidad de productos de cada categoría, ordenado de mayor a menor cantidad. El reporte debe tener la siguiente forma:

category_name	count(*) 1
Storage	108
CPU	70
Mother Board	60
Video Card	50

c) Un cliente a nivel mundial requiere saber qué bodegas de almacenamiento (warehouses) tiene la compañía y dónde se encuentran ubicadas. Para eso, le han solicitado un listado de todos los países junto a sus respectivas ubicaciones de almacenes, ordenado alfabéticamente por país. El reporte debe tener la siguiente forma:

country_name 1	warehouse_name	address	city	state
Canada	Bombay	147 Spadina Ave	Toronto	Ontario
Italy	Southlake, Texas	1297 Via Cola di Rie	Roma	*NULL*
Italy	San Francisco	93091 Calle della Testa	Venice	*NULL*
Japan	Seattle, Washington	9450 Kamiya-cho	Hiroshima	*NULL*
Japan	New Jersey	2017 Shinjuku-ku	Tokyo	Tokyo Prefecture
United States of America	Sydney	2011 Interiors Blvd	South San Francisco	California
United States of America	Beijing	2004 Charade Rd	Seattle	Washington
United States of America	Toronto	2014 Jabberwocky Rd	Southlake	Texas
United States of America	Mexico City	2007 Zagora St	South Brunswick	New Jersey

d) El área de ventas que cubre la ciudad de San Francisco, requiere un listado actualizado con los inventarios de los productos ‘Intel’ que existen en el almacén ‘San Francisco’. Genere un reporte con la cantidad de inventario de productos ‘Intel’ existentes en dicho almacén, ordenado alfabéticamente por producto. El reporte debe tener la siguiente forma:

product_id	product_name ▲ 1	quantity
120	Intel Core 2 Extreme QX9775	183
71	Intel Core i7-3930K	155
27	Intel Core i7-3960X Extreme Edition	106
73	Intel Core i7-4770K	111
200	Intel Core i7-4790K	176
132	Intel Core i7-5930K	185
214	Intel Core i7-5960X	196
19	Intel Core i7-6950X (OEM/Tray)	101
69	Intel Core i7-7820X	154
198	Intel Core i7-980	175
210	Intel Core i9-7900X	194
144	Intel DG43RK	95

e) El gerente general de la compañía desea conocer el valor total del inventario (inventario valorizado) existente en cada almacén. Para llevar a cabo dicho requerimiento, confeccione un reporte que acumule el costo de cada producto multiplicado por la cantidad, agrupado por almacén y ordenado de forma descendente por dicho valor. El reporte debe tener la siguiente forma:

warehouse_name	inventario_valorizado ▼ 1
San Francisco	18743692.54
Sydney	13065912.82
Seattle, Washington	11338582.45
Toronto	8873312.49
New Jersey	8306634.66
Beijing	8223708.39
Southlake, Texas	5841509.94
Bombay	5741230.61
Mexico City	5692248.50

2. Construir un algoritmo de chequeo de margen de venta de un producto

Un objetivo importante de este proyecto es solucionar el problema actual en el cual los vendedores muchas veces aplican descuentos excesivos a los productos para cerrar una venta y así cumplir con las metas mensuales.

Para esto, se decidió crear un algoritmo que realice el chequeo de cada uno de los ítems que componen una orden de compra y aplique ciertas reglas de negocio para detectar si se está haciendo un buen o mal uso de los descuentos.

Esta es una idea que surgió recientemente pero que aún la gerencia comercial no está del todo convencida que dará el resultado esperado. Es por este motivo que el arquitecto ha decidido que lo mejor es construir una pieza de software extensible, con comportamiento polimórfico, para que a futuro se puedan incorporar nuevas lógicas de chequeo disminuyendo el impacto en el software.

Se le solicita que desarrolle un algoritmo de chequeo de margen de un producto con comportamiento polimórfico. Construya dos algoritmos, uno que realice un **cálculo simple** y otro que realice un **cálculo complejo**.

El cálculo del margen de venta se realiza de la siguiente manera:

$$Margen = \frac{PrecioVenta - Costo}{PrecioVenta}$$

Por ejemplo, si un producto se vende a 150 y tiene un costo de 100, el margen de ventas es de (150-100)/150 = 0,33, que es lo mismo que un 33,3%.

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

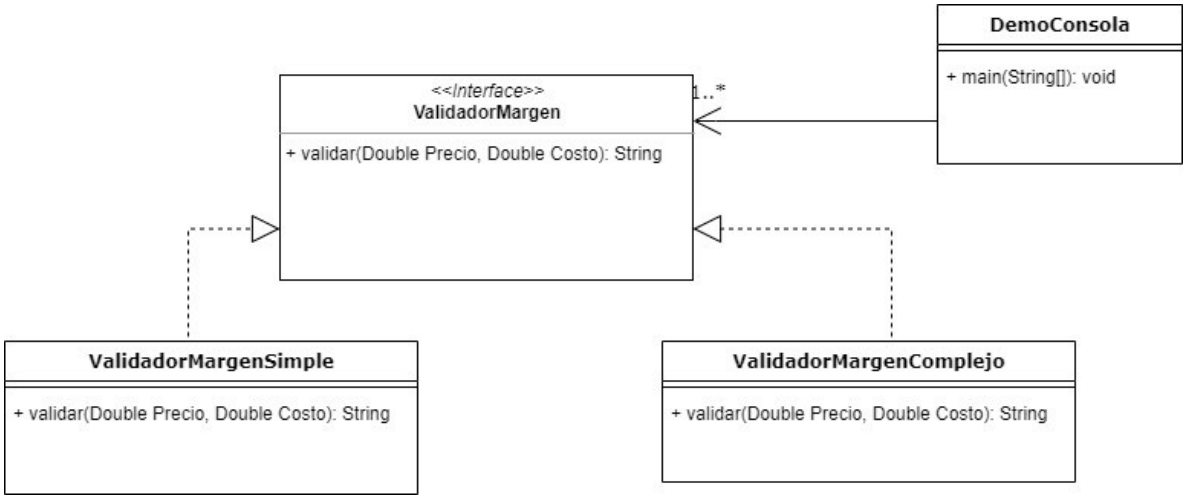
- El algoritmo deberá recibir el precio de venta y el costo del producto, para realizar el cálculo del margen.
- Si el margen de un producto es mayor o igual a 10%, entonces el algoritmo deberá retornar “MARGEN OK”
- Si el margen de un producto es menor al 10%, entonces el algoritmo debe retornar “ALERTA MARGEN BAJO MINIMO”

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

- El algoritmo deberá recibir el precio de venta y el costo del producto, para realizar el cálculo del margen.
- Si el margen de un producto es mayor o igual a 10%, entonces el algoritmo deberá retornar “MARGEN OK”
- Si el margen de un producto es menor al 10%, entonces el algoritmo debe retornar “ALERTA MARGEN BAJO MINIMO”
- Si el margen de un producto es menor que 0, entonces el algoritmo deberá retornar “ERROR MARGEN NEGATIVO”
- Si el margen de un producto es superior a 100%, entonces deberá retornar “ADVERTENCIA POSIBLE ERROR EN COSTO”, siempre y cuando el precio del producto sea igual o superior a 1.000 US$

- En el caso de un producto de menos de 1.000 US$, entonces deberá lanzar “ADVERTENCIA POSIBLE ERROR COSTO” si es que el margen es superior a 50%.

Ambos algoritmos deben recibir el valor del precio y del costo, y deberán retornar una cadena de caracteres con el mensaje. 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 valores aleatorios, tanto para el precio como para el costo, entre 200 y 2000, y que posteriormente realice la validación del margen utilizando las dos implementaciones creadas anteriormente (algoritmo simple y complejo). Repita este cálculo 5 veces.

A continuación, se presenta un ejemplo de la ejecución:

```
----------------------------------------
Demostración Algoritmo Verificación Margen
----------------------------------------

1. Precio 345.02 y Costo 106.65
Algoritmo Simple: MARGEN OK
Algoritmo Complejo: ADVERTENCIA POSIBLE ERROR EN COSTO

2. Precio 1535.67 y Costo 341.92
Algoritmo Simple: MARGEN OK
Algoritmo Complejo: ADVERTENCIA POSIBLE ERROR EN COSTO

3. Precio 641.45 y Costo 752.24
Algoritmo Simple: ALERTA MARGEN BAJO MINIMO
Algoritmo Complejo: ERROR MARGEN NEGATIVO

4. Precio 1254.98 y Costo 1190.45
Algoritmo Simple: ALERTA MARGEN BAJO MINIMO
```

```
Algoritmo Complejo: ALERTA MARGEN BAJO MINIMO

5. Precio 701.16 y Costo 598.
Algoritmo Simple: MARGEN OK
Algoritmo Complejo: MARGEN OK
```

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 valores aleatorios para ir invocando a cada algoritmo.

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 algoritmo de validación del margen de venta de productos

Construya una clase de pruebas en Java que permita verificar el correcto funcionamiento del algoritmo de “cálculo complejo” de validación del margen de venta, considerando al menos los siguientes tests:

- Tests que considere casos normales, con distinta cantidad y valores de stock.
- Tests que considere condiciones de borde, por ejemplo, qué pasa cuando valores de borde, u otra condición de excepción.

4. Crear Monitor de Productos

Para mejorar la gestión de los vendedores de terreno, se requiere crear una página web dinámica que despliegue el Monitor de Productos, tal como se detalla en la siguiente imagen mock-up.

Se pide:

a) Desplegar el listado de almacenes en el primer combobox, ordenado alfabéticamente, con valores que provengan de la base de datos
b) Desplegar el listado de categorías en el segundo combobox, ordenado alfabéticamente, con valores que provengan de la base de datos
c) Desplegar el listado de los productos y su inventario (de acuerdo al detalle especificado en la imagen mock-up) de acuerdo a los filtros seleccionados (tienda y categoría), a partir de la información que se encuentra en la base de datos.

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

- Utilizar jsp y taglibs jstl para el despliegue de la vista (u otra tecnología de vista)
- Utilizar bootstrap para los elementos
- Que los elementos se ajusten a distintos tamaños de pantalla

5. Crear una API REST que disponibilice el monitor de productos

Disponibilice un servicio REST que permita obtener la misma información del Monitor de Productos. Recuerde que el servicio podría recibir como la tienda y categoría.