FUNCIONALIDAD:

El controlador (clase ControladorTienda) se encarga de gestionar la interacción entre la vista (clase InterfazTienda) y las implementaciones DAO para los modelos Producto y Fabricante, a través de las respectivas instancias de cada clase (fabDAO y prodDAO).

La vista se encarga de:

Reproducir la interfaz en la que figuran las tablas de productos y fabricantes, como así también una barra de navegación con las distintas acciones de las que dispone el usuario como…

1 – Creación de productos y fabricantes.

2 – Actualización de productos y fabricantes.

3 – Eliminación de productos y fabricantes.

4 – Búsqueda de productos y fabricantes.

El controlador se encarga de hacer de mediador entre la vista y la base de datos, de modo que la interacción con la base se vea más abstraída.

Los modelos se encargan de dar forma a un nuevo registro o permitir acceder de forma más cómoda a aquellos devueltos por una consulta.

VALIDACIÓN:

Se implementó la validación en la interfaz, la cual arroja advertencias al momento de encontrar algún fallo en el ingreso de datos (como el ingreso de un campo vacío o el de un valor inesperado en un campo de tipo Double). En casos puntuales, como el de la modificación o eliminación de registros, se lanzarán excepciones personalizadas de intentar operar en un registro inexistente (lo cual se determina a través del nombre ingresado) o de intentar modificar el nombre de un registro existente por otro también presente en la base de datos.

De hallar valores nulos se asume que el usuario ha cancelado la operación, por lo que se le ofrece la opción de regresar al menú (el frame donde figuran tanto tablas como barra de navegación).

INFORME:

* Por momentos el código retornaba un CommunicationsException al realizar la conexión a base de datos, cosa la cual se solucionaba al ir a servicios e iniciar manualmente MySQL.
* También tuve inconvenientes al querer cerrar los recursos de un ResultSet o los distintos Statement al haber cerrado previamente la conexión, soltaba una excepción. Busqué información y resulta que los demás recursos se cierran junto a la conexión, por lo que prescindí de los métodos destinados a limpiar recursos y me limité a cerrar la conexión.
* En el método actualizar, si el usuario ingresaba un nuevo nombre y este ya existía en la base de datos se arrojaba una excepción y un mensaje de éxito cuando no se había modificado el registro. Por tanto fue preciso añadir una sentencia que corroborara que el nombre proporcionado no estuviese presente peeeero entonces ahí arrojaba error si el nombre era el original, por lo que hubo que añadir otra sentencia con la que se hiciera caso omiso si el producto era el original.

Y pim, pum, pam surgió otro problema.

Este por mensa, y es que había asumido que el método equals se aplicaba al operador == por lo que Java de forma implícita haría uso del método que yo hubiese sobreescrito para tal fin. Pero pues no, de modo que usé el equals. Pero con el equals proporcionado por defecto por NetBeans pasaron cosas también, por lo que lo retoque y no me pregunten cómo, ahí anduvo.

* También hubieron otros inconvenientes, como en la búsqueda, en el método del DAO (encargado de devolver un listado con todas las coincidencias) no manejé la posibilidad de que, de ser un listado vacío, desde fuera no se lo tomaría por nulo (que era lo que esperaba la vista recibir en caso de que no existieran coincidencias) por lo que opté por un ternario en el return (si el tamaño es mayor a 0, devuelve el listado, sino un null) y asunto resuelto.

Seguro y podría seguir pero me compadezco de quién pueda llegar a leer esto sin cabecear en el proceso. La conclusión es feliz: Anda!

(Favor de no mirar con lupa igual)