SHOES SHOP

SISTEMA GESTION ZAPATERÍA

Integrantes:

- Cristaldo Gustavo
- Florio Fatima
- Heredia Alexis
- Rotela Kevin
- Tucci Elias

Desarrollaremos el sistema de una zapatería, en donde se distingue el tipo de calzado según su uso.

Tendremos las clases:

Test:

1)obtenerListaDeCalzadosDelCliente()

Este test, asegura que los calzados que el cliente decidió comprar sean los mismos que están registrados. Verificaremos según el tamaño de la lista, el Size y también, el stock de cada producto, que sea coincidente con el que se vendió.

Métodos surgidos del test:

- agregarCalzado(calzado, cantidadDeCalzados): busca en la lista de calzados de la tienda si existe el calzado, si no existe, añade el calzado y le setea la cantidad que ingresa, es decir el stock; si existe, solo incrementa el stock a el mismo.
- venderCalzado(cliente, calzadoUno, cantidadAVender, empleado): lo que hace es buscar por id el calzado a vender, si existe, le descuenta lo que se va a vender, si no existe, o la cantidad a vender supera al stockDisponible, devuelve false.

Al mismo tiempo, si existe y se pudo descontar la cantidad de la tienda (Se pudo vender), creamos la clase ClienteCalzado, que se forma de un cliente y un calzado y de la cantidad de pares de calzados. Esto es para la relación 1 cliente tiene muchos calzados y viceversa.

Buscamos en la lista de clientesCalzados si existe el clienteCalzado, si existe y tiene el mismo cliente y mismo calzado idéntico, le aumenta el stockCliente y la cantidadAComprar a ese calzado del clienteCalzado, sino crea un nuevo ClienteCalzado y lo agrega a la lista.

El stockCliente es un atributo de calzado que separa al stock original que es con el que trabaja la tienda del stockCliente que es la cantidad que compró el cliente.
Luego cuando quiera ver la lista de zapatos de cierto cliente, buscaré en clientesCalzados y si coincide el cliente, voy agregando a una lista nueva de calzados los getCalzado() del cliente y devuelvo esa lista. La lista de clientesCalzados podría quedarme por ejemplo:
Cliente1 – Calzado Botin 23 – stockCliente: 1
Cliente1 – Calzado Botin 25 – stockCliente: 1
Cliente2 – Calzado OutDoor 20 – stockCliente: 1
Cliente3 – Calzado Running 26 – stockCliente: 2

Luego llega la parte de EmpleadoCliente, que registra los clientes a los que cierto empleado pudo venderles. Es similar a la relación clienteCalzado. El EmpleadoCliente está formado por estos dos y la cantidad que se vendio. Para luego calcular la cantidadDePares que vendio cierto empleado. Entonces busca en la lista de empleadosClientes si coincide algún objeto y si se realizó la venta, le incrementa la cantidad vendida. Sino crea un nuevo objeto de cero. Luego podremos ver los clientes de cierto empleado y el total de ventas.

- De este test saldrán los test:

queSePuedanAgregarCalzadosALaTienda , queSePuedaVenderCalzado
-El stock estará oculto. Al añadir a la tienda calzados, yo le digo cuántos voy a añadir y esto setea el stock de este calzado, entonces me añade por ejemplo, un botin cuyo stock nuevo es 6 pares.

2) obtenerLosCalzadosPorClienteOrdenadosPorPrecioDescendente ()

En este test, haremos que se ingresen calzados a una tienda, se vendan a un cliente, y luego pueda obtener la lista de calzados que el cliente compro ordenada por precio descendiente, es decir, del más caro al más barato.
Será como el test 1) solo que debo corroborar que la lista tenga un orden especifico No usaré HashSet o TreeSet porque no me permiten duplicados.
Debo hacer un método similar a obtenerlistaDeZapatosDeCliente(cliente) pero con orden porPrecioDescendiente

3)obtenerListaDeClientesDelEmpleado()

En este tes chequeamos que al venderse calzados a cierto cliente podamos obtener la lista de calzados adquiridos por cierto cliente. De aquí sale el método obtenerListaDeClientesDeEmpleado(empleado) que busca en empleadosClientes de la tienda a cierto empleado y si coincide va guardando su el cliente en una nueva lista de clientes.

4) public void obtenerElTotalDeVentasPorEmpleado()

En este test se venden calzados y al vender cierta cantidad, si se puede vender se le suma a empleadoCliente la cantidad que este empleado va vendiendo a cada cliente. Esto hace el método obtenerTotalDeVentasTotalesDeEmpleado(empleado). Si coincide el empleado te da la cantidad vendida.

5) public void queSePuedaAnadirCalzadoBotinALaTienda()

En este test se busca comprobar que se pueda añadir correctamente un calzado de tipo botín a nuestra tienda. Para esto utilizamos el método crearBotin(), el cual nos ayuda a crear una nueva instancia de nuestro objeto y un asserTrue que verificara que la adición se realizo correctamente.

6) public void queSePuedaAnadirCalzadoRunningALaTienda()
En este test se busca comprobar que se pueda añadir correctamente un calzado de tipo running a nuestra tienda. Para esto utilizamos el método crearRunning(), el cual nos ayuda a crear una nueva instancia de nuestro objeto y un asserTrue que verificara que la adición se realizó correctamente.

7) public void queSePuedaAnadirCalzadoOutDoorALaTienda()

En este test se busca comprobar que se pueda añadir correctamente un calzado de tipo botín a nuestra tienda. Para esto utilizamos el método crearOutDoor(), el cual nos ayuda a crear una nueva instancia de nuestro objeto y un asserTrue que verificara que la adición se realizo correctamente.

8) public void queSePuedaCrearElCliente()

En este test se busca comprobar que se pueda crear correctamente un cliente.

9) public void queSePuedaCrearElEmpleado()

En este test se busca comprobar que se pueda crear correctamente un empleado.

10)public void calcularSueldoTotalSegunTipoEmpleado()

Este test se encarga de comprobar que se realice de mandera exitosa la asignacion del sueldo a un empleado tomando en cuenta su rubro, disposicion horaria y categoria.

11) public void buscarCalzadoPorCodigo()

Este test se encarga de verificar que se retorne el calzado de la lista mediante su ID.

12) public void buscarCalzadoPorCodigoDevuelvePrecio()

Este test se encarga de verificar que, si el calzado se encuentra registrado, se retorne su precio.

13) public void buscarCalzadoTipoOrdenadoPorTalle()

Este test se encarga de verificar que se ordene a cada tipo de calzado (botin, outdoor y running) según el talle de manera ascendente siempre y cuando los mismos estén registrados en la tienda.

14) public void asignarComisionASueldo()

Este test se encarga de comprobar que se le dé un bonus al sueldo base del empleado teniendo en cuenta su antigüedad, la categoría y disponibilidad horaria del mismo.

15) public void QueTraigaTodosLosRunning ()
Este test primeramente agrega varios calzados, con la función agregarcalzado(), para agregar variedad al array de calzado, método donde se envía por parámetro al calzado creado, que puede ser repetido (por eso se lo busca por ID), también manda el stock, que en caso de ya existir, le agrega el stock al que ya tenía. Ahora sí no existe, y es la primera vez q se agrega al array de calzados, mediante el add se lo agrega y se setea el stock que parte desde cero en el constructor.
Una vez que se agregan, por el método ObtenerTodosLosRunning() se verifica que se traiga un array local calzado, donde solo están las instancias de calzado buscadas, por el instance of que se encarga de filtrar eso.

16) public void QueTraigaTodosLosOutDoor ()

Este test primeramente agrega varios calzados, con la función agregarcalzado(), para agregar variedad al array de calzado, método donde se envía por parámetro al calzado creado, que puede ser repetido (por eso se lo busca por ID), también manda el stock, que en caso de ya existir, le agrega el stock al que ya tenía. Ahora sí no existe, y es la primera vez q se agrega al array de calzados, mediante el add se lo agrega y se setea el stock que parte desde cero en el constructor.
Una vez que se agregan, por el método ObtenerTodosLosOutDoor() se verifica que se traiga un array local calzado, donde solo están las instancias de calzado buscadas, por el instance of que se encarga de filtrar eso.

17) public void QueTraigaTodosLosBotin ()

Este test primeramente agrega varios calzados, con la función agregarcalzado(), para agregar variedad al array de calzado, método donde se envía por parámetro al calzado

creado, que puede ser repetido (por eso se lo busca por ID), también manda el stock, que en caso de ya existir, le agrega el stock al que ya tenía. Ahora sí no existe, y es la primera vez q se agrega al array de calzados, mediante el add se lo agrega y se setea el stock que parte desde cero en el constructor.
Una vez que se agregan, por el método ObtenerTodosLosBotin() se verifica que se traiga un array local calzado, donde solo están las instancias de calzado buscadas, por el instance of que se encarga de filtrar eso.

18) public void calcular comisión del empleado()

En este test, se agrega el empleado en cuestión, y después se calcula la comisión: para esto, tomamos como parámetro para calcular esta comisión la antigüedad y la categoría, también si es Full-Time o Part-time, poniendo un valor por año de trabajo

19) public void asignarCalzadosAUnCliente()

En este test se busca comprobar que, al momento de realizar una venta, el calzado quede relacionado con el comprador, armando así una lista con el historial de compras.
20) public void asignarClienteAUnEmpleado()

En este test se busca comprobar que, al momento de realizar una venta, el vendedor quede relacionado con el cliente, armando así una lista con el historial de ventas realizadas.

21) public void calcularElTotalGastadoPorCadaCliente()

En este test se verifica que se pueda calcular de manera exitosa el total gastado por cada cliente, basándose en la lista creada anteriormente en donde se almacena su historial de compras

22) public void venderCalzadoAUnCliente()

En este test comprobamos que se pueda vender un calzado a un cliente y que, al suceder esto, el stock, la lista de ventas del empleado y la lista de comprados de cliente sufran sus respectivas modificaciones.