

## Trabajo Práctico N°6 Pruebas Unitarias con JUnit

3- Responda las siguientes preguntas:

a) ¿Por qué no se dibuja la relación de agregación entre la clase `CollectionProducto` y `Producto`?

La relación de agregación está representada en el diagrama de manera simplificada. Se puede ver la multiplicidad en el atributo "`Productos : Producto [1..*]`" de la clase `CollectionProducto` indica que esta contiene una colección de objetos `Producto`. Aunque no se emplea una flecha para ver su conexión, la semántica de agregación sigue siendo evidente.

b) ¿Cuántos atributos tiene la clase `Factura`? ¿Cuáles son los nombres de estos atributos?

La clase `Factura` tiene dos atributos:

1- `fecha`: Un atributo de tipo `LocalDate`, que representa la fecha de la factura.

2- `nroFactura`: Un atributo de tipo `long`, que representa el número único de cada factura.

c) ¿Cómo se llama la relación que se establece entre `Factura` y `Detalle`?

La relación entre `Factura` y `Detalle` es de composición. Esto significa que una `Factura` está compuesta por uno o más `Detalles`, los cuales no pueden existir independientemente de la `Factura`. Si se elimina la `Factura`, los detalles asociados también se eliminan.

d) ¿Cómo se llama la relación entre las clases `Factura` y `Cliente`?

La relación entre `Factura` y `Cliente` es de asociación. Una `Factura` está asociada a un `Cliente`, indicando que la factura pertenece a un cliente específico, pero el cliente puede existir independientemente de la factura. Es una relación donde un cliente puede tener varias facturas.

e) ¿Por qué los atributos de las clases `Collections` son públicos?

Se asume que los atributos de las clases "`Collection`" (`CollectionProducto`, `CollectionCliente`, etc.) se encuentran privados o protegidos, siguiendo las buenas prácticas de encapsulamiento, normalmente para acceder a estos atributos, se utilizarían métodos públicos tales como `getters`, `setters`, o métodos específicos diseñados para manipular las colecciones.

f) Describa las características de todos los métodos de la clase `CollectionClientes`.

La clase `CollectionClientes` tiene los métodos:

1- `buscarCliente(in dni: long): Cliente`: Busca un cliente en la colección utilizando su DNI como clave, devuelve el objeto `Cliente` buscado si lo encuentra, en caso contrario retorna un valor nulo.

2- `agregarCliente(in cliente: Cliente)`: Este método permite agregar un nuevo objeto `Cliente` a la colección de clientes.

3- `precargarClientes()`: Este método se encarga de cargar clientes iniciales para no tener que crear cada uno de los clientes.