Ejercicio Agregación

Andres Felipe Urrego Rodriguez

Julio Roberto Galvis Cardozo

Servicio Nacional de Aprendizaje

Centro de diseño y metrología

Tecnólogo en Análisis y Desarrollo de Software

2848938

2024

Tabla de contenido

1. Introducción

En el ámbito de la programación orientada a objetos, las clases se pueden relacionar de diversas maneras. Estas relaciones permiten modelar cómo los objetos interactúan y colaboran entre sí dentro de un sistema. En este informe, se aborda la relación de asociación, centrándonos en la relación de agregación, que es una forma más específica de asociación. Se utiliza un proyecto de UML desarrollado en Netbeans como ejemplo para ilustrar este concepto.

2. Relación de Asociación

Una relación de asociación es una conexión estructural entre dos o más clases. Representa cómo los objetos de estas clases se vinculan para intercambiar información o colaborar en diversas tareas. Dependiendo del tipo de asociación, el nivel de dependencia y colaboración entre los objetos puede variar.

3. Relación de Agregación

La agregación es un tipo específico de asociación. En esta relación, una clase posee una propiedad de otra clase. Cuando un objeto de la clase contenida es pasado como parámetro al constructor de la clase contenedora, decimos que esa clase está "agregada". Esto significa que el objeto agregado persiste incluso después de que finalice el ámbito del objeto contenedor.

Por ejemplo, en un sistema de facturación, una factura puede agregar productos. Los productos siguen existiendo, aunque la factura se elimine.

4. Proyecto UML en NetBeans

El proyecto UML utilizado como ejemplo es un subsistema que permite la creación de comprobantes, tales como facturas y remitos, que contienen información sobre el cliente, los productos y el total de la compra. En el diagrama de clases, se visualizan varias entidades: Comprobante, Factura, Cliente, y Producto.

En este sistema, se observa que:

La clase Comprobantes generaliza a la clase Factura.

Los comprobantes tienen un tipo ("F" para Factura, "R" para Remito), un número correlativo y una fecha.

- El ejemplo destaca cómo los productos se agregan a las facturas mediante una relación de agregación con multiplicidad. Esto significa que en una factura puede haber uno o más productos.

5. Diagrama de Clases: Comprobantes

En el diagrama de clases presentado en el proyecto, Comprobantes generaliza a Factura. Los comprobantes tienen tipos específicos, y cada comprobante incluye información sobre el cliente y los productos.

La relación entre Factura y Producto es de agregación, permitiendo que múltiples productos se asocien con una factura. Además, existe una relación entre Factura y Cliente, donde cada factura está relacionada con un único cliente.

6. Agregación con y sin Navegabilidad

Existen dos tipos de agregación:

Agregación sin navegabilidad: En este caso, aunque una clase contiene a otra, no se necesita establecer una propiedad explícita en la clase contenedora. La clase contenedora simplemente utiliza los objetos de la clase agregada sin la necesidad de definirlos como propiedades.

Agregación con navegabilidad: Aquí, la clase contenedora debe definir explícitamente una propiedad que referencia a la clase agregada, permitiendo navegar desde la clase contenedora hacia la clase agregada.

- En el diagrama, los símbolos que representan estas relaciones son:
- Un rombo vacío que indica agregación sin navegabilidad.
- Un rombo con una flecha para la agregación con navegabilidad.

7. **Multiplicidad**

La multiplicidad define cuántos objetos pueden estar involucrados en la relación de agregación. Por ejemplo:

- La relación entre Factura y Cliente tiene una multiplicidad de 1 a 1, lo que significa que una factura solo puede estar asociada a un cliente.
- La relación entre Factura y Producto tiene una multiplicidad de 1 a muchos (1. *), permitiendo que una factura incluya uno o más productos.

8. Conclusión

La relación de agregación es un concepto clave en la programación orientada a objetos, ya que permite modelar la interacción entre objetos de una manera estructurada. En sistemas como el descrito, las clases se relacionan entre sí de forma que una clase puede contener y administrar objetos de otras clases, pero estos objetos mantienen su independencia. El uso de UML y herramientas como Netbeans facilita la representación gráfica de estas relaciones, lo que contribuye a una mejor comprensión y organización del sistema.