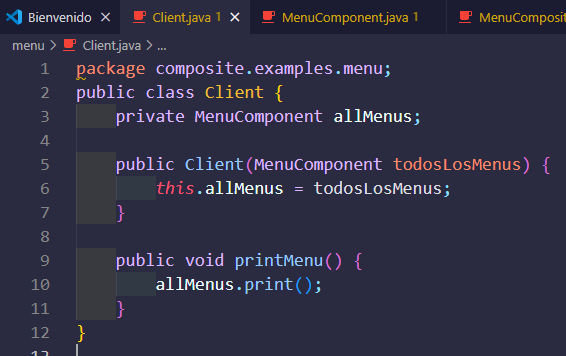


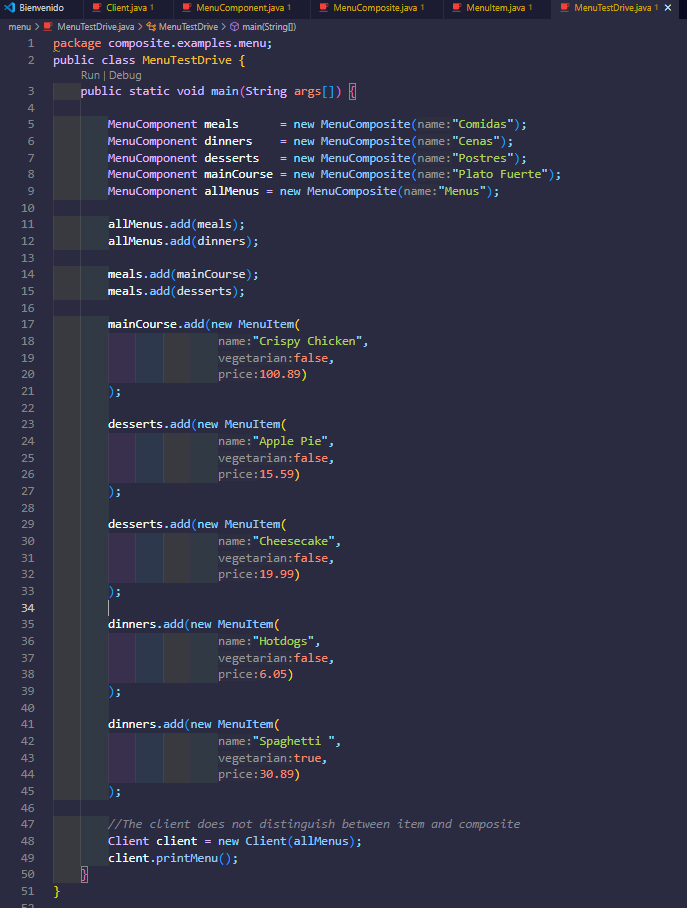
**Codigo GitHub**

****

****





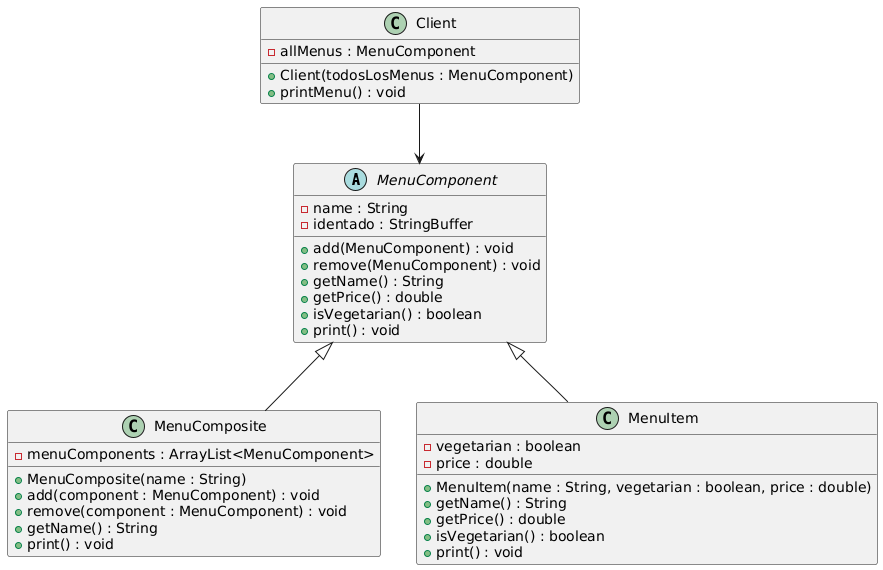


### Análisis del Código y Uso del Patrón Composite

1. **Clases Clave**:
   * **MenuComponent**: Es la clase base abstracta, que define la estructura común para los elementos y las composiciones de menús, incluyendo métodos como add(), remove(), y print(), los cuales son sobrescritos en subclases según sea necesario. Métodos como getPrice() y isVegetarian() lanzan una excepción en esta clase base porque pueden no aplicarse a todos los componentes (e.g., un menú de varios elementos no tiene un precio único).
   * **MenuComposite**: Esta clase representa una composición de MenuComponent (es decir, un grupo de componentes). Implementa los métodos add(), remove(), y print() para gestionar y mostrar todos los componentes de forma jerárquica.
   * **MenuItem**: Representa un elemento del menú que tiene atributos específicos como price y vegetarian. Implementa los métodos getPrice(), isVegetarian() y print() para mostrar sus propios detalles.
2. **Cliente (Client)**: El cliente trabaja con la jerarquía de MenuComponent, llamando al método print() en allMenus. Dado que allMenus puede contener tanto elementos individuales (MenuItem) como compuestos (MenuComposite), no necesita diferenciar entre ellos; simplemente llama a print() y el sistema determina la lógica.

### Diferencia con un Enfoque Tradicional de POO

En un enfoque tradicional de POO, cada tipo de objeto (e.g., un menú o un elemento de menú) podría tener métodos únicos, y el cliente tendría que conocer la clase exacta de cada objeto para manipularlo adecuadamente. Sin el patrón Composite, el cliente necesitaría realizar verificaciones condicionales para determinar si está trabajando con un menú o un elemento antes de llamarlos, lo que resultaría en un código más complejo y difícil de mantener. El uso del patrón Composite permite evitar este problema al ofrecer una interfaz uniforme.

**DIAGRAMA UML**

**Explicación del Diagrama UML**

**MenuComponent: Clase abstracta que define la interfaz común para todos los componentes. Las clases MenuComposite y MenuItem heredan de esta clase.**

**MenuComposite: Esta clase es una colección de MenuComponent y, por lo tanto, puede agregar o eliminar componentes.**

**MenuItem: Representa elementos individuales en el menú, como platos específicos, con atributos adicionales (price y vegetarian).**

**Client: Contiene una referencia al componente raíz MenuComponent y llama a printMenu() para mostrar el menú.**