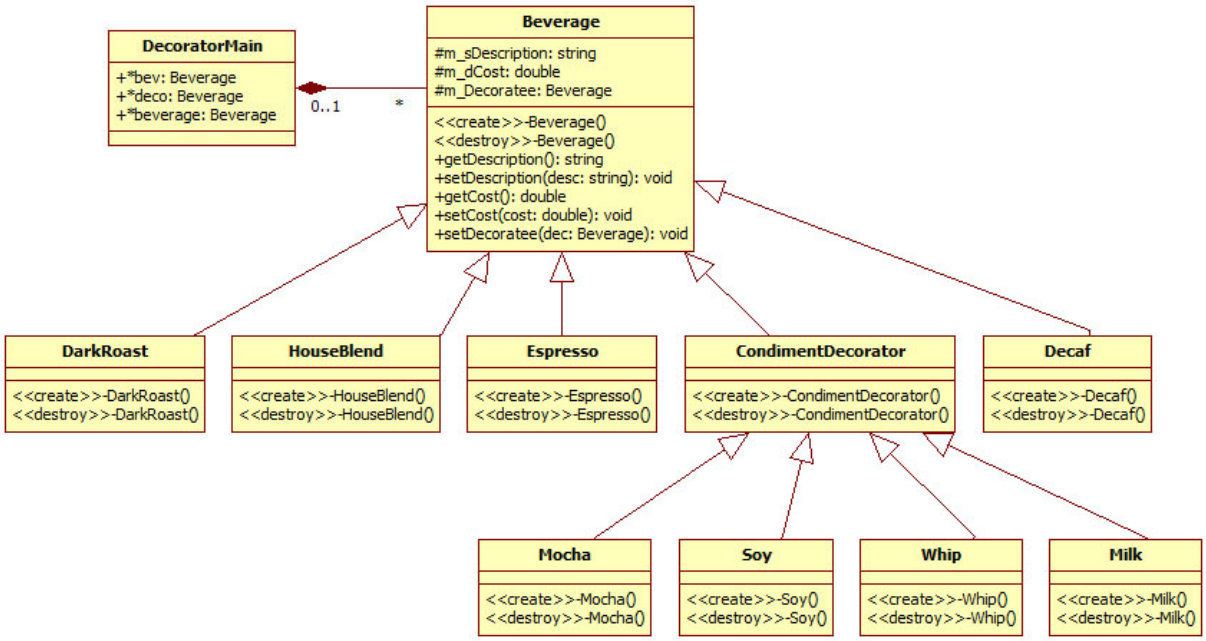
Decorator Design Pattern

GoF Statement:	**Attaches additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.**
Category:	**Structural**
UML Diagram:	



Description of the Demonstration:

A parent class, Beverage, has four main sub-classes, DarkRoast, HouseBlend,Espresso, and Decaf. These represent the four types of beveages available from the Starbuzz Coffee Shop. Another sub-class, CondimentDecorator, serves as the parent class for all condiments to be added to the beverages following the Decorator Design Pattern. In the demonstration three "customers" place orders, giving first the Beverage type they want, e.g. "I'll have a DarkRoast" then the condiments they want, e.g. "with milk and mocha". The Beverage is then successively decorated with each of the condiments, e.g. the instance of Milk is given a pointer to an instance of DarkRoast, then an instance of Mocha is given a pointer to the instance of Milk. A call can then be made to getCost on the Mocha which calls getCost on Milk which call getCost on Darkroast. The final result gives the total cost of this Beverage. A similar chain of calls is used to print the description of the Beverage.