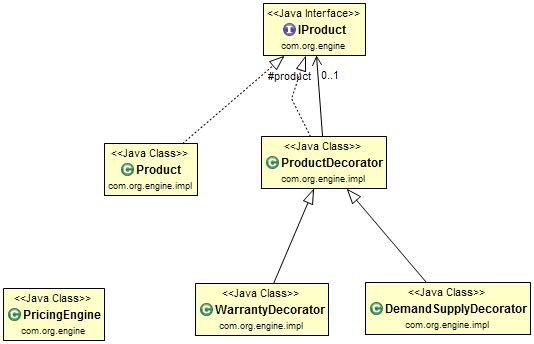
**Approach for the Pricing Engine**



1. To support multiple parameters for the product, Decorator Design Pattern is used.
2. Entry point for the user to test is the Pricing Engine
3. IProduct (Component Interface)

Is an interface which has a method name calculate.

1. Product Class (Component Implementation) implement IProduct

Perform the task of finding the chosen price depending on the Pricing Strategy entered by the user (Least/Highest)

1. ProductDecorator (Decorator) which implements IProduct
2. DemandSupplyDecorator(Concrete Decorator) extends ProductDecorator

Perform the task of find the recommened price (by taking the chosen price from Product Class) depending on the user input.

1. WarrantyDecorator(Concrete Decorator) extends ProductDecorator

Perform the task of find the recommened price (by taking the chosen price from Product Class) depending on the user input.

1. If the user wants to add new parameters for the product then he just need to make a new Concrete Decorator class just like the WarrantyDecorator.

**Assumptions:**

1. Depending on the Parameters entered by the user for the product the recommended price will change.
2. So for the mock purpose I have made a Warranty as a parameter with following conditions

If Warranty support needed for product, Product is sold 5 % more than chosen price.

If Warranty support not needed for product, Product is sold 5 % less than chosen price.