**Problem statement:**   
  
Design class structure for controller of a Coffee brewing machine. The controller should dispense beverages : latte, cappuccino, espresso, mocha on the press of respective buttons.

Ex:

espresso = Coffee + hot water

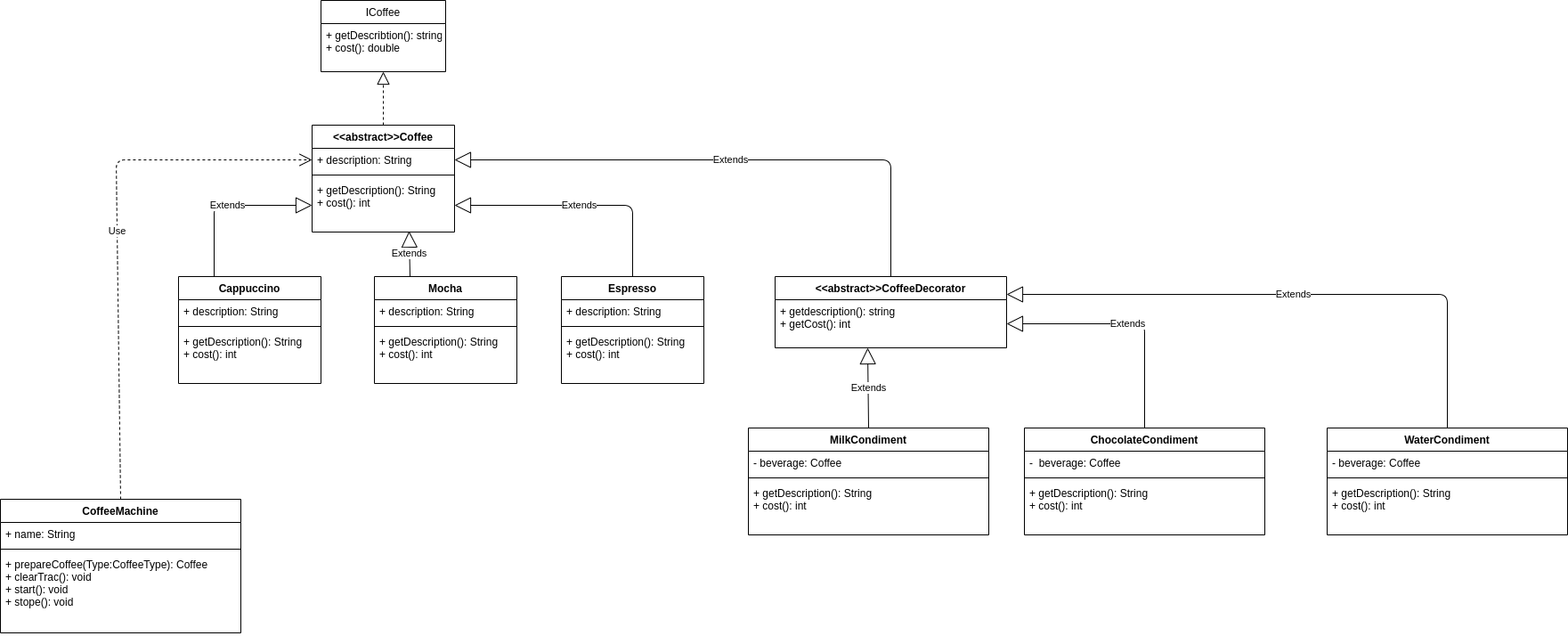
latte = Coffee + hot water + milk

Note: Each Coffee machine should be configurable to dispense one or more beverages from the list independently.

**Design pattern(s) used:**   
  
I used Decorator design pattern.  
  
**Design Justification :**

I used decorator design pattern because as per given problem we need to create an coffee instance and wanted to Attach additional responsibilities to an object dynamically. So decorator provides a flexible to add subclasses for extending the funcationality.

**Design Representation :**



**PseudoCode : classes and interface  
  
  
Icoffee:**{ getDescription(), cost() }

**abstract Coffee implement Icoffee** { //implements method} **abstract CoffeeDecorator extends Coffee** { declared extended method as abstract}  
  
**Espressio extends Coffee:** {override extended coffee class method}

**Latte extends Coffee**: {override extended coffee class method}

**Codiment classes:  
milk extends coffeeDecorator** { override the method}

**water** **extends coffeeDecorator** { override the method}

**Chocolate extends coffeeDecorator** { override the method}

**CoffeeMachine:** { //has serval methods and calling differnet coffee creation objects.}