**题目：**某饮料店卖饮料时，可以根据顾客的要求在饮料中加入各种配料，饮料店会根据顾客所选的饮料种类和所加入的配料来计算价格。饮料店所供应的部分饮料及配料的种类和价格如图中所示。请你选择恰当的设计模式，为该饮料店开发一个计算饮料价格的程序，要求给出设计思路，画出类图，并写出关键代码。



**解答：**

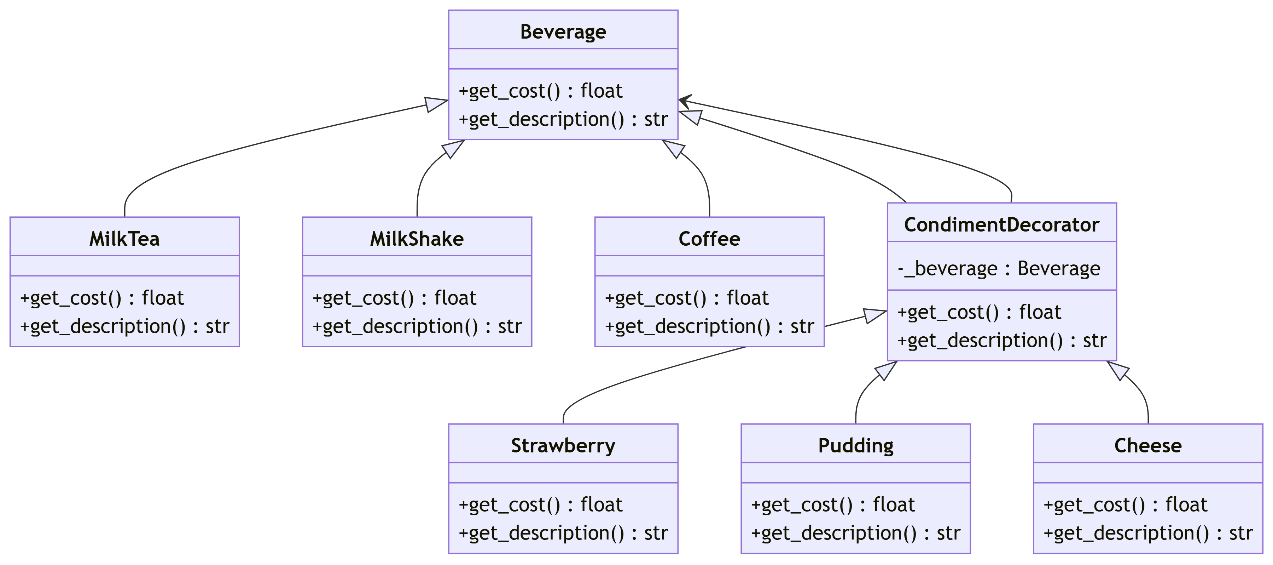
* **设计思路：**

1. 定义一个抽象的饮料类，包含获取饮料价格和描述的基本方法。
2. 奶茶、奶昔和咖啡，继承自饮料的基类，实现饮料价格和描述方法。
3. 每种配料作为装饰者，动态地为饮料对象增加额外的功能，如价格和描述，从而实现配料的附加。

* **模式：**

装饰模式

* **类图：**



|  |
| --- |
| * **核心代码：** |
| from abc import ABC, abstractmethod  *# 饮料的抽象基类*  class Beverage(ABC):      @abstractmethod      def get\_cost(self) -> float:          pass      @abstractmethod      def get\_description(self) -> str:          pass  *# 奶茶类*  class MilkTea(Beverage):      def get\_cost(self) -> float:          return 8.0      def get\_description(self) -> str:          return "Milk Tea"  *# 奶昔类*  class MilkShake(Beverage):      def get\_cost(self) -> float:          return 9.0      def get\_description(self) -> str:          return "Milk Shake"   咖啡类  class Coffee(Beverage):      def get\_cost(self) -> float:          return 10.0      def get\_description(self) -> str:          return "Coffee"  *# 配料装饰的基类*  class CondimentDecorator(Beverage):      def \_\_init\_\_(self, beverage: Beverage):          self.\_beverage = beverage  *# 草莓装饰*  class Strawberry(CondimentDecorator):      def get\_cost(self) -> float:          return self.\_beverage.get\_cost() + 2.0      def get\_description(self) -> str:          return self.\_beverage.get\_description() + ", Strawberry"  *# 布丁装饰*  class Pudding(CondimentDecorator):      def get\_cost(self) -> float:          return self.\_beverage.get\_cost() + 3.0      def get\_description(self) -> str:          return self.\_beverage.get\_description() + ", Pudding"  *# 芝士装饰*  class Cheese(CondimentDecorator):      def get\_cost(self) -> float:          return self.\_beverage.get\_cost() + 4.0      def get\_description(self) -> str:          return self.\_beverage.get\_description() + ", Cheese"  *# 先选择一杯奶茶*  beverage = MilkTea()  print(beverage.get\_description(), "￥", beverage.get\_cost())  *# 添加草莓配料*  beverage = Strawberry(beverage)  print(beverage.get\_description(), "￥", beverage.get\_cost()) |