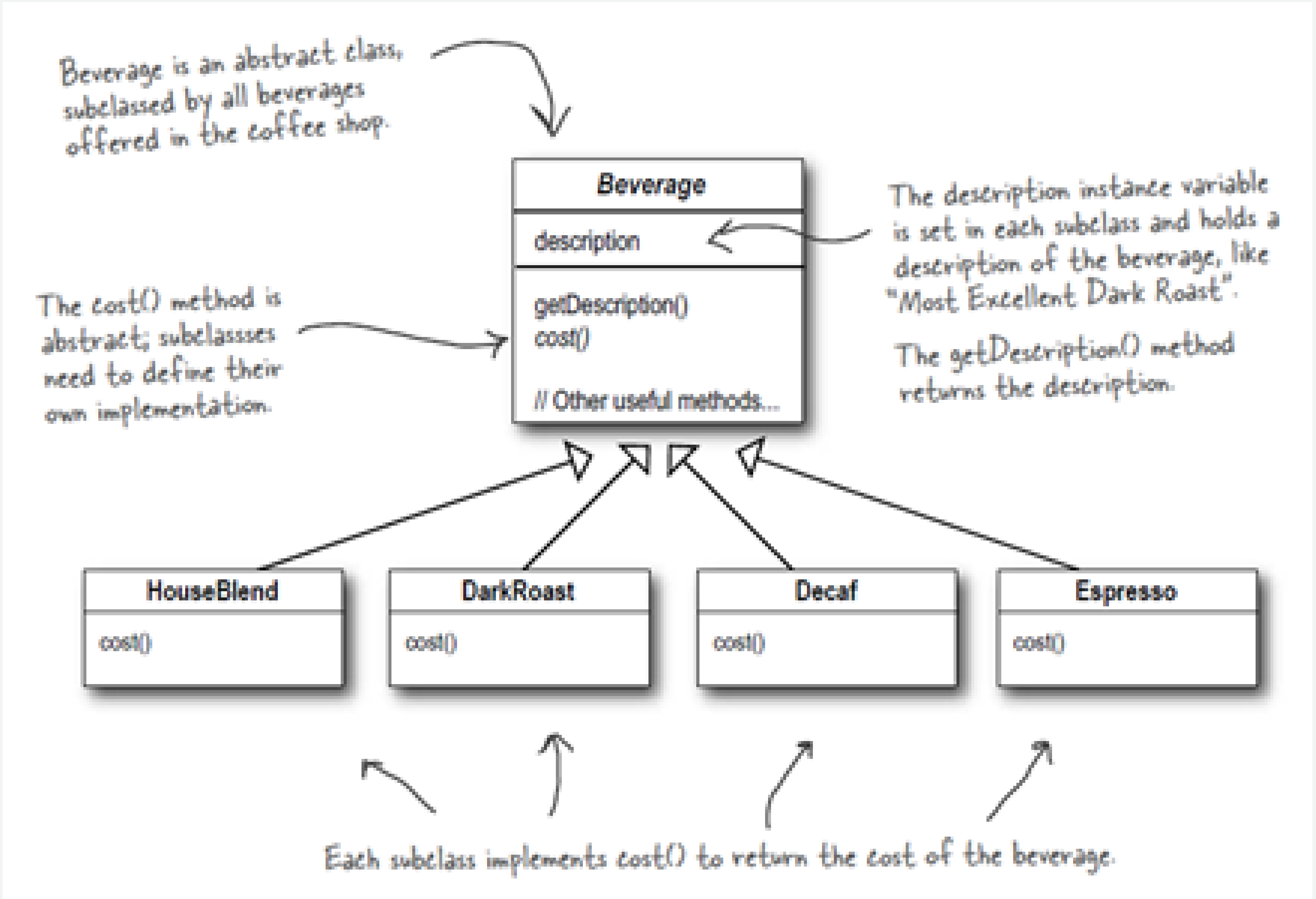
THE DECORATOR PATTERN

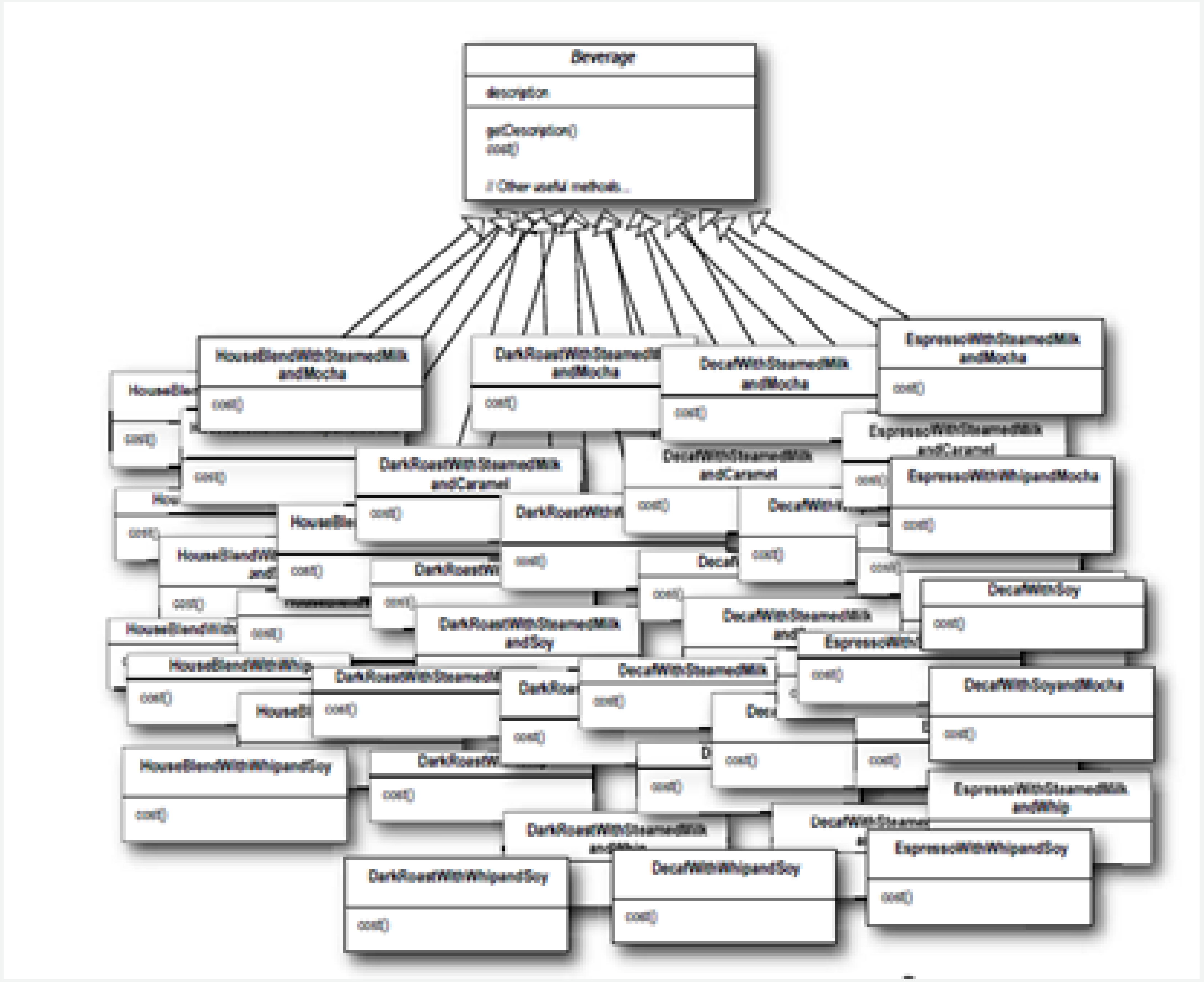
TRẦN HỮU BÁCH - ĐỖ DUY HIỆP

COFFEE MENU
AMERICANO
MACHIATTO
ESPRESSO
CAFFE LATTE
OPEN

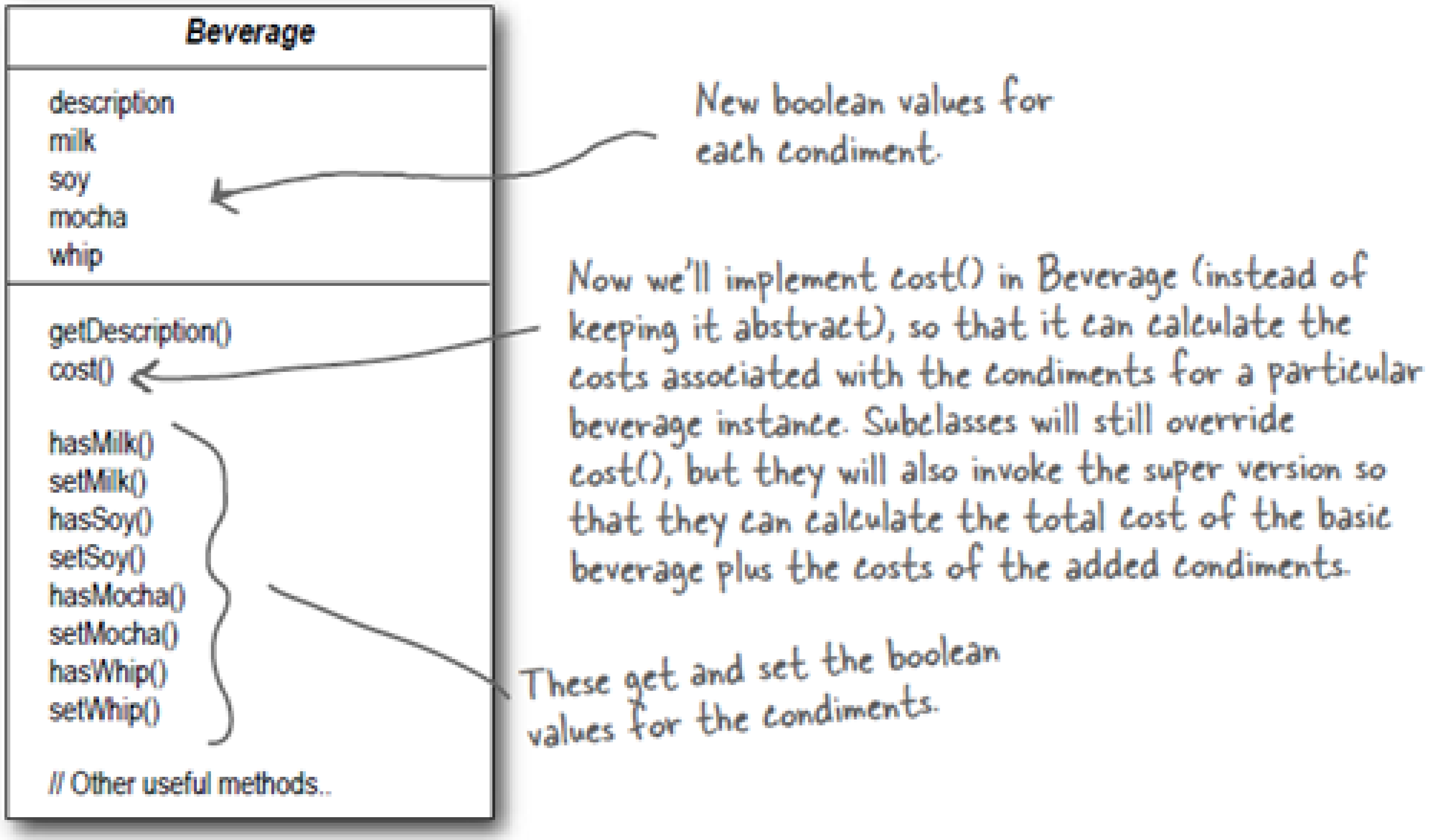
SCHEMA FOR THE COFFEE SHOP


Beverage is an abstract class, subclassed by all beverages offered in the coffee shop.
Beverage
description
getDescription()
cost()
// Other useful methods...
The description instance variable is set in each subclass and holds a description of the beverage, like "Most Excellent Dark Roast".
The getDescription() method returns the description.
The cost() method is abstract; subclassses need to define their own implementation.
HouseBlend
cost()
DarkRoast
cost()
Decaf
cost()
Espresso
cost()
Each subclass implements cost() to return the cost of the beverage.

PROBLEM

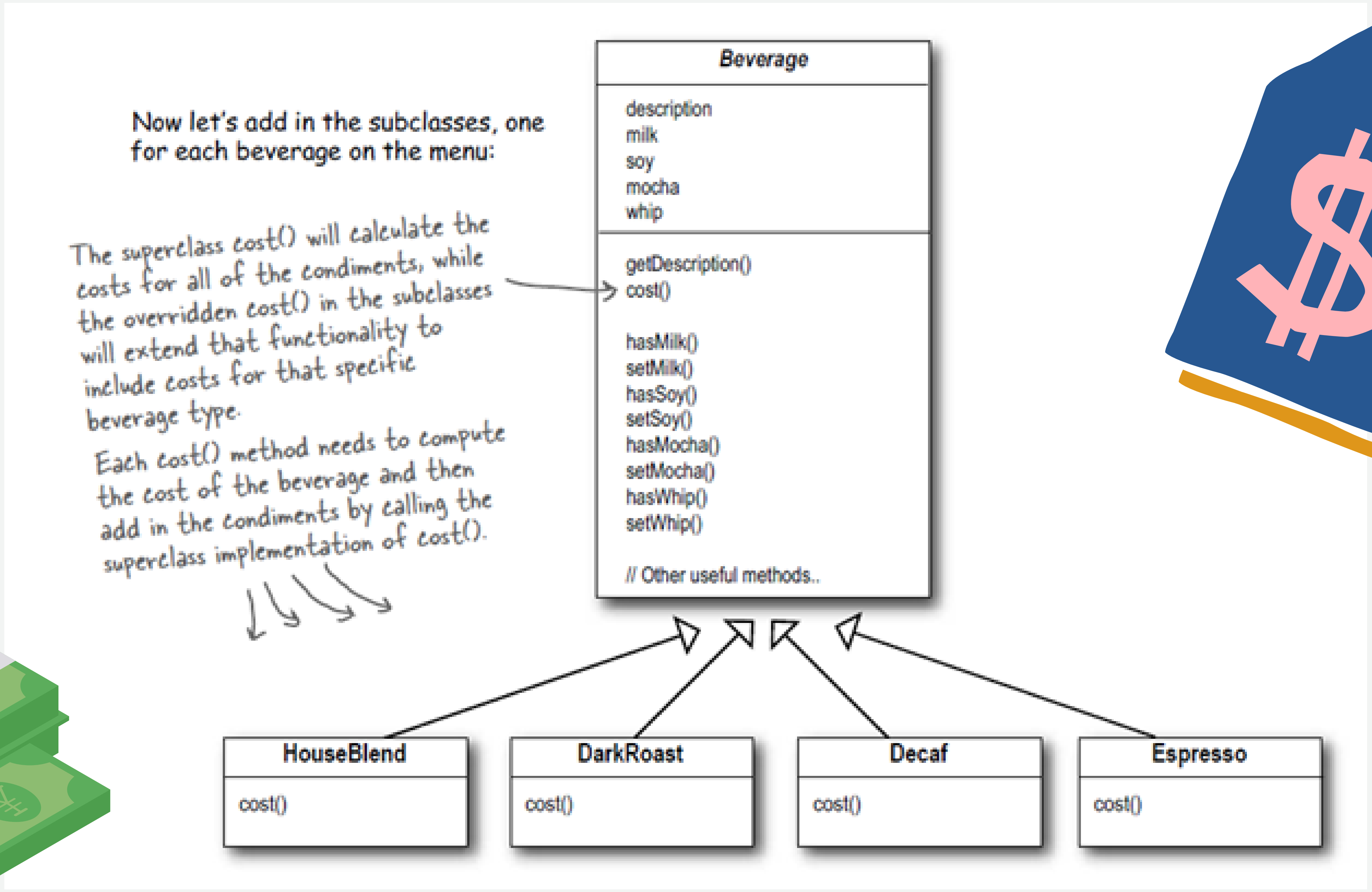

Beverage
description
getDescription()
cost()
// Other useful methods...
HouseBlendWithSteamedMilk andMocha
DarkRoastWithSteamedMilk andMocha
DecafWithSteamedMilk andMocha
EspressoWithSteamedMilk andMocha
EspressoWithSteamedMilk andCaramel
EspressoWithWhipandMocha
DarkRoastWithSteamedMilk andCaramel
DecafWithSteamedMilk andCaramel
DecafWithSoy
DarkRoastWithSteamedMilk andSoy
DecafWithSoyandMocha
HouseBlendWithWhipandSoy
EspressoWithSteamedMilk andWhip
DarkRoastWithSteamedMilk andWhip
DarkRoastWithWhipandSoy
DecafWithWhipandSoy
EspressoWithWhipandSoy
cost()

FIRST IDEA



Beverage
description
milk
soy
mocha
whip
getDescription()
cost()
hasMilk()
setMilk()
hasSoy()
setSoy()
hasMocha()
setMocha()
hasWhip()
setWhip()
// Other useful methods..
New boolean values for each condiment.
Now we'll implement cost() in Beverage (instead of keeping it abstract), so that it can calculate the costs associated with the condiments for a particular beverage instance. Subclasses will still override cost(), but they will also invoke the super version so that they can calculate the total cost of the basic beverage plus the costs of the added condiments.
These get and set the boolean values for the condiments.

COMPUTE COST


Now let's add in the subclasses, one for each beverage on the menu:
The superclass cost() will calculate the costs for all of the condiments, while the overridden cost() in the subclasses will extend that functionality to include costs for that specific beverage type.
Each cost() method needs to compute the cost of the beverage and then add in the condiments by calling the superclass implementation of cost().
Beverage
description
milk
soy
mocha
whip
getDescription()
cost()
hasMilk()
setMilk()
hasSoy()
setSoy()
hasMocha()
setMocha()
hasWhip()
setWhip()
// Other useful methods..
HouseBlend
cost()
DarkRoast
cost()
Decaf
cost()
Espresso
cost()

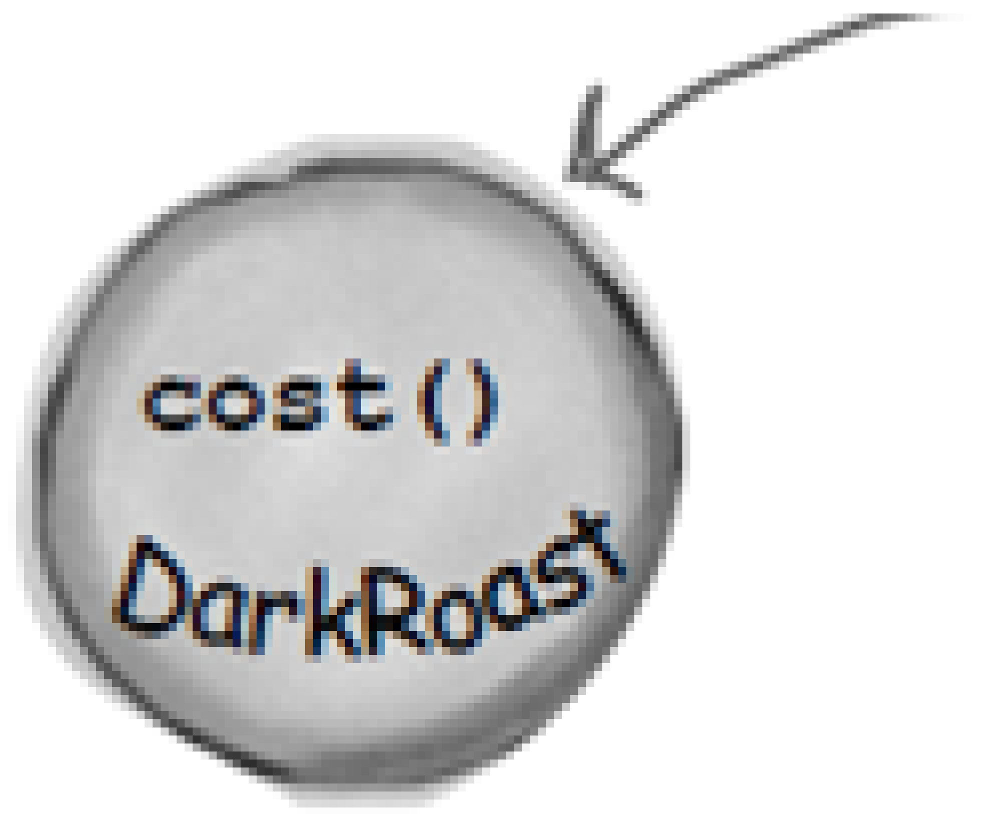
COMPUTE COST

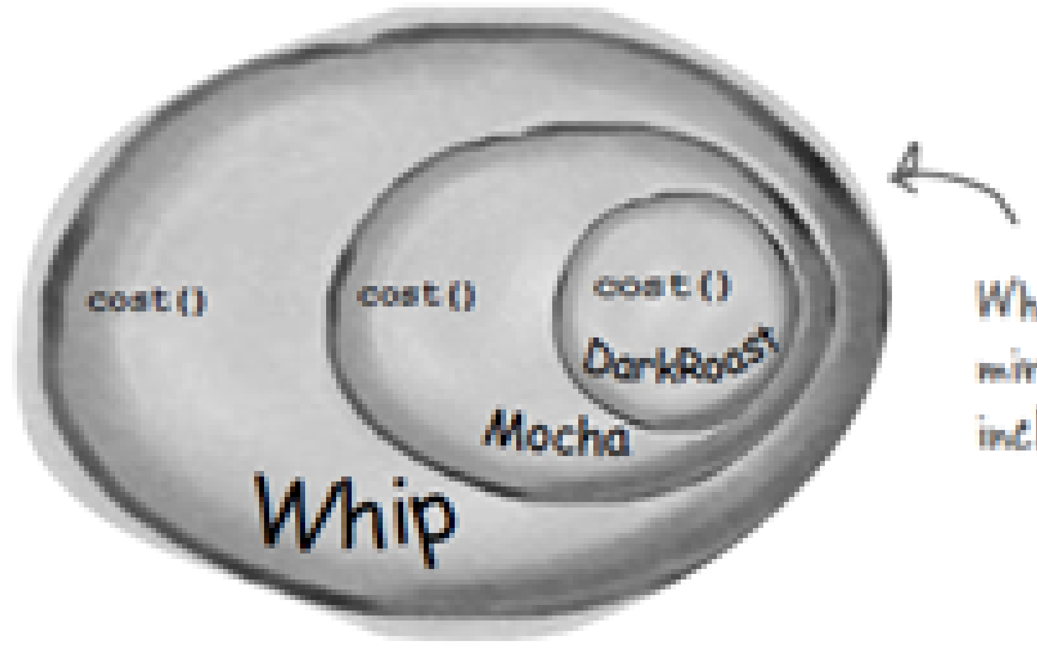
0.99$

2.99$

2$

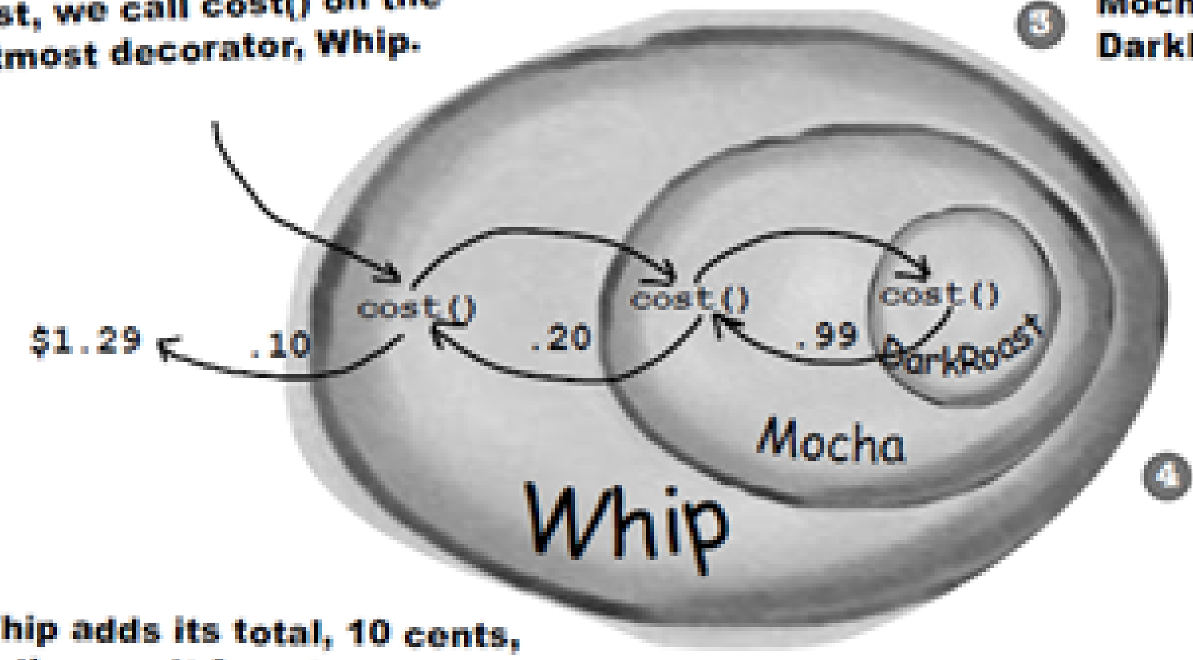
BEVERAGE AND "DECORATE" IT WITH THE CONDIMENTS





Whip is a
mirrors D
includes a cost() method.

So, a DarkRoast wrapped in Mocha and Whip is still a Beverage and we can do anything with it we can do with a DarkRoast, including call its cost() method.



2 Whip calls cost() on Mocha.

a few pages.

1 First, we call cost() on the outmost decorator, Whip.

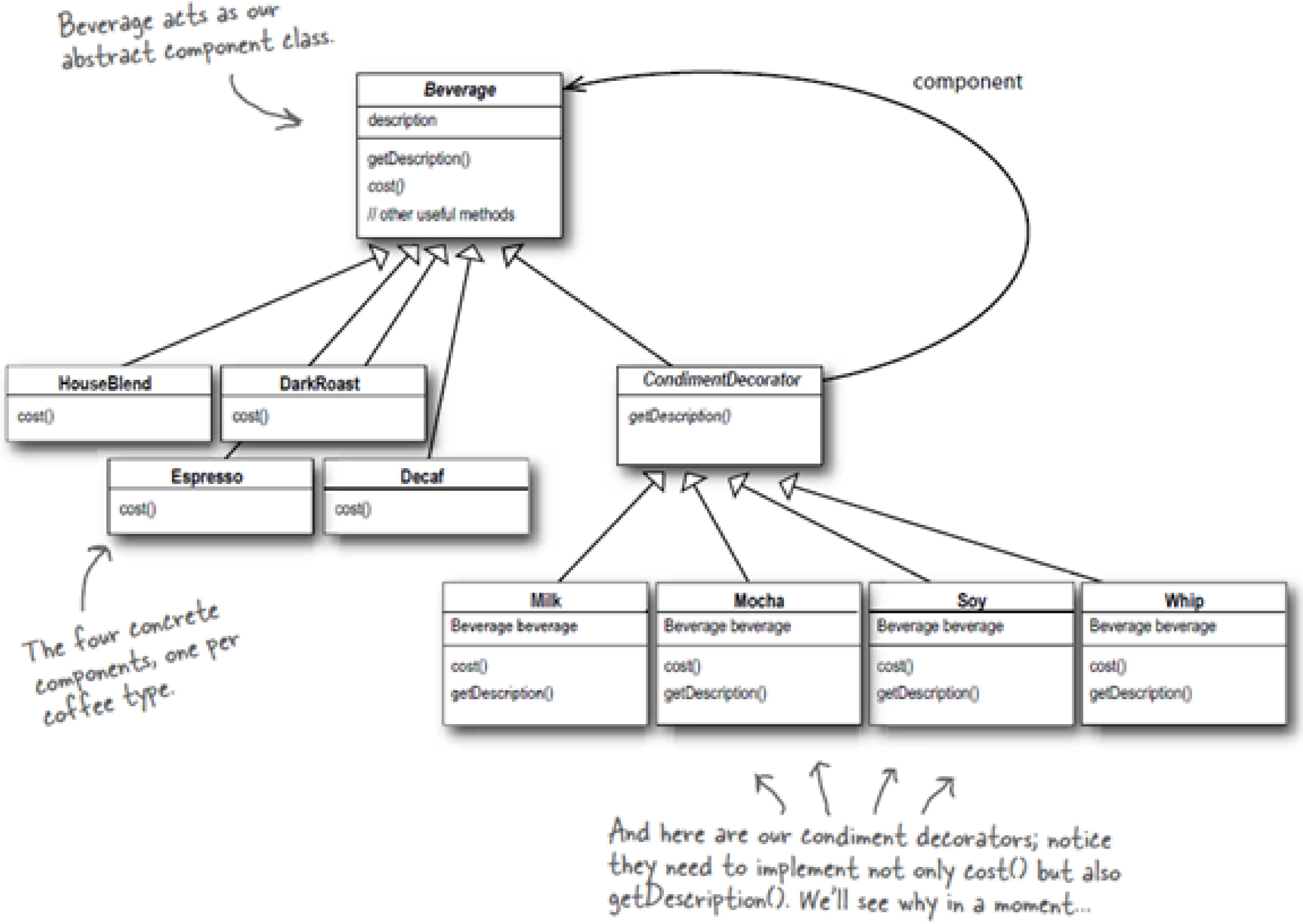
3 Mocha calls cost() on DarkRoast.

4 DarkRoast returns its cost, 99 cents.

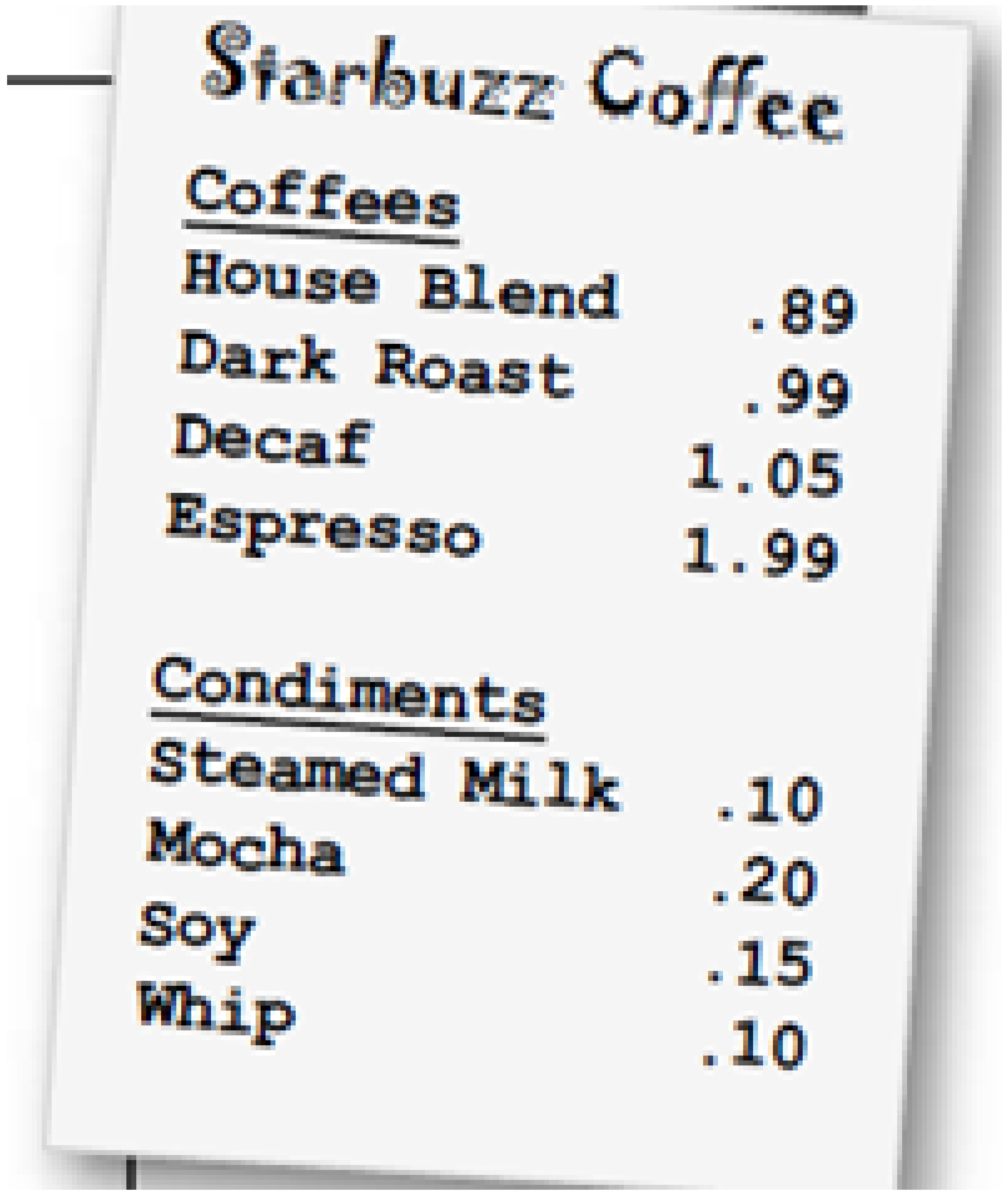
5 Whip adds its total, 10 cents, to the result from Mocha, and returns the final result—$1.29.

5 Mocha adds its cost, 20 cents, to the result from DarkRoast, and returns the new total, $1.19.

THE DECORATOR PATTERN



OUR MENU



Starbuzz Coffee

Coffees

House Blend	.89
Dark Roast	.99
Decaf	1.05
Espresso	1.99

Condiments

Steamed Milk	.10
Mocha	.20
Soy	.15
Whip	.10