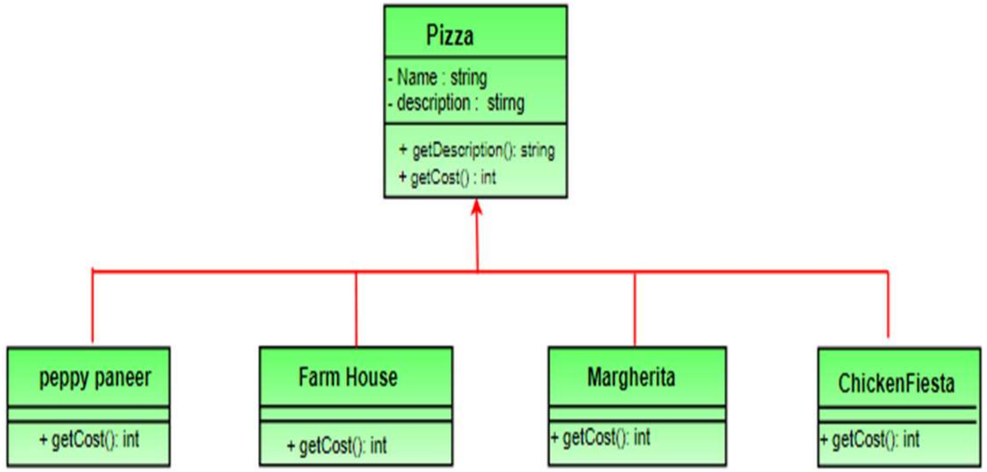
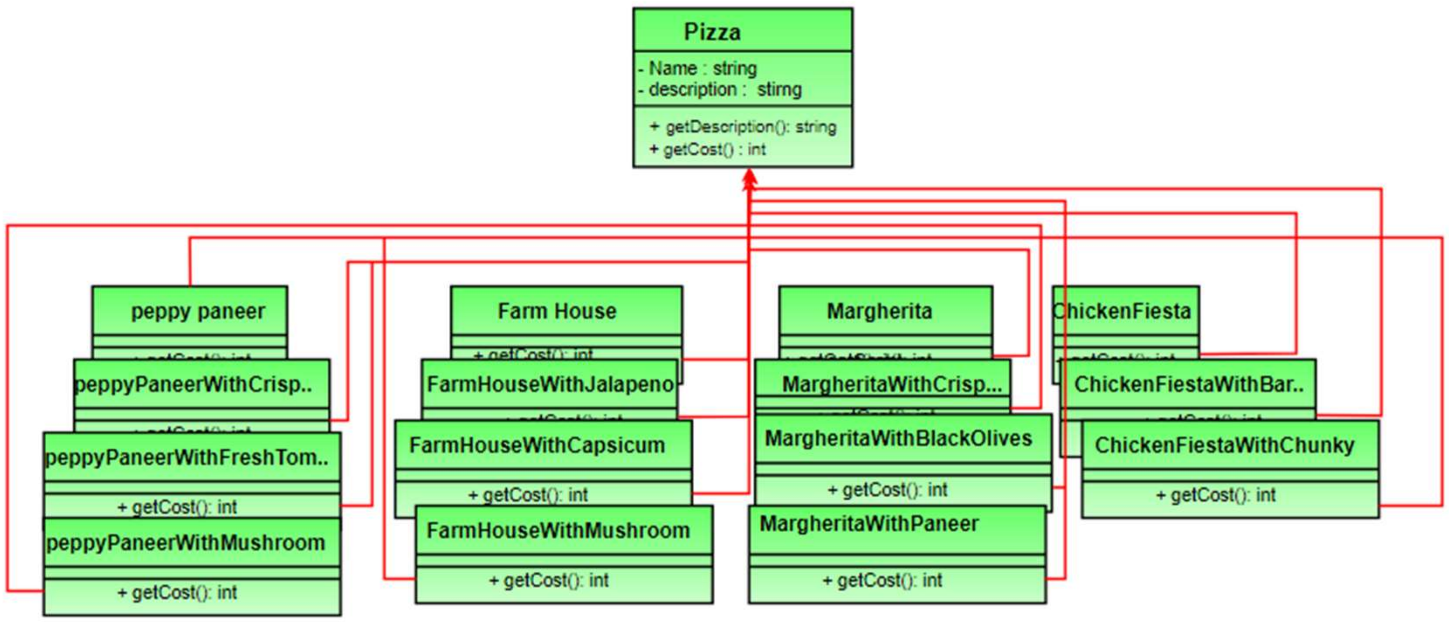
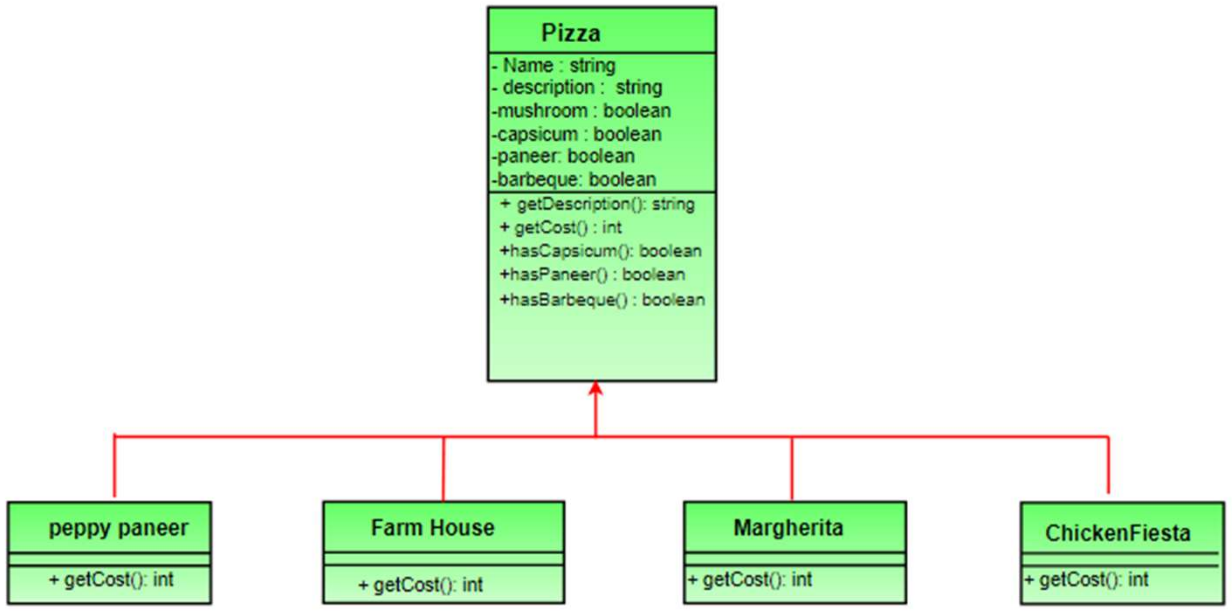
PIZZA USE CASE

- Suppose we are building an application for a pizza store and we need to model their pizza classes. Assume they offer four types of pizzas namely Peppy Paneer, Farmhouse, Margherita and Chicken Fiesta. Initially we just use inheritance and abstract out the common functionality in a Pizza class.




Pizza
- Name : string
- description : stirng
+ getDescription(): string
+ getCost() : int
peppy paneer
peppyPaneerWithCrisp..
peppyPaneerWithFreshTom..
+ getCost(): int
peppyPaneerWithMushroom
+ getCost(): int
Farm House
+ getCost(): int
FarmHouseWithJalapeno
FarmHouseWithCapsicum
+ getCost(): int
FarmHouseWithMushroom
+ getCost(): int
Margherita
MargheritaWithCrisp...
MargheritaWithBlackOlives
+ getCost(): int
MargheritaWithPaneer
+ getCost(): int
ChickenFiesta
ChickenFiestaWithBar..
ChickenFiestaWithChunky
+ getCost(): int

OPTION 1



Pizza
- Name : string
- description : string
-mushroom : boolean
-capsicum : boolean
-paneer: boolean
-barbeque: boolean
+ getDescription(): string
+ getCost() : int
+hasCapsicum(): boolean
+hasPaneer() : boolean
+hasBarbeque() : boolean
peppy paneer
+ getCost(): int
Farm House
+ getCost(): int
Margherita
+ getCost(): int
ChickenFiesta
+ getCost(): int

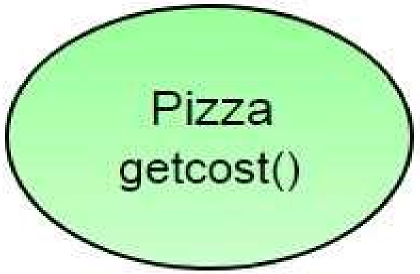
OPTION 2

OPTION 2 - CONTINUED

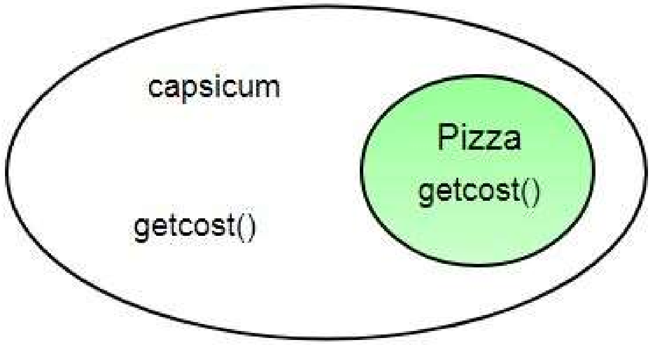
- This design looks good at first but lets take a look at the problems associated with it.
  - Price changes in toppings will lead to alteration in the existing code.
  - New toppings will force us to add new methods and alter getCost() method in superclass.
  - For some pizzas, some toppings may not be appropriate yet the subclass inherits them.
  - What if customer wants double capsicum or double cheeseburst?
- In short our design violates one of the most popular design principle – **The Open-Closed Principle** which states that classes should be open for extension and closed for modification.

DECORATOR DESIGN PATTERN

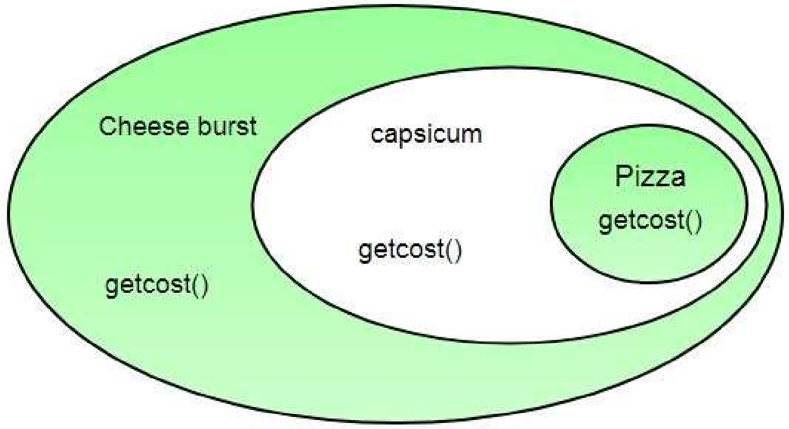
- Take a pizza object.



- “Decorate” it with a Capsicum object.



- “Decorate” it with a CheeseBurst object
- What we get in the end is a pizza with cheeseburst and capsicum toppings. Visualize the “decorator” objects like wrappers.

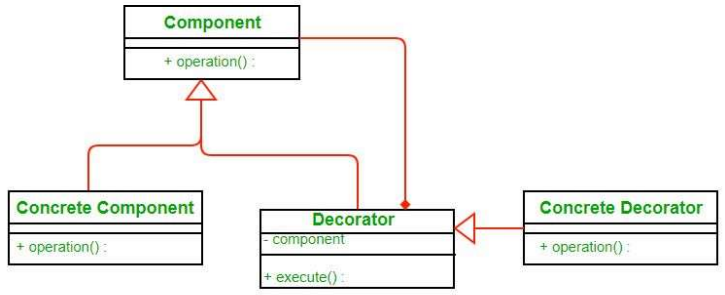


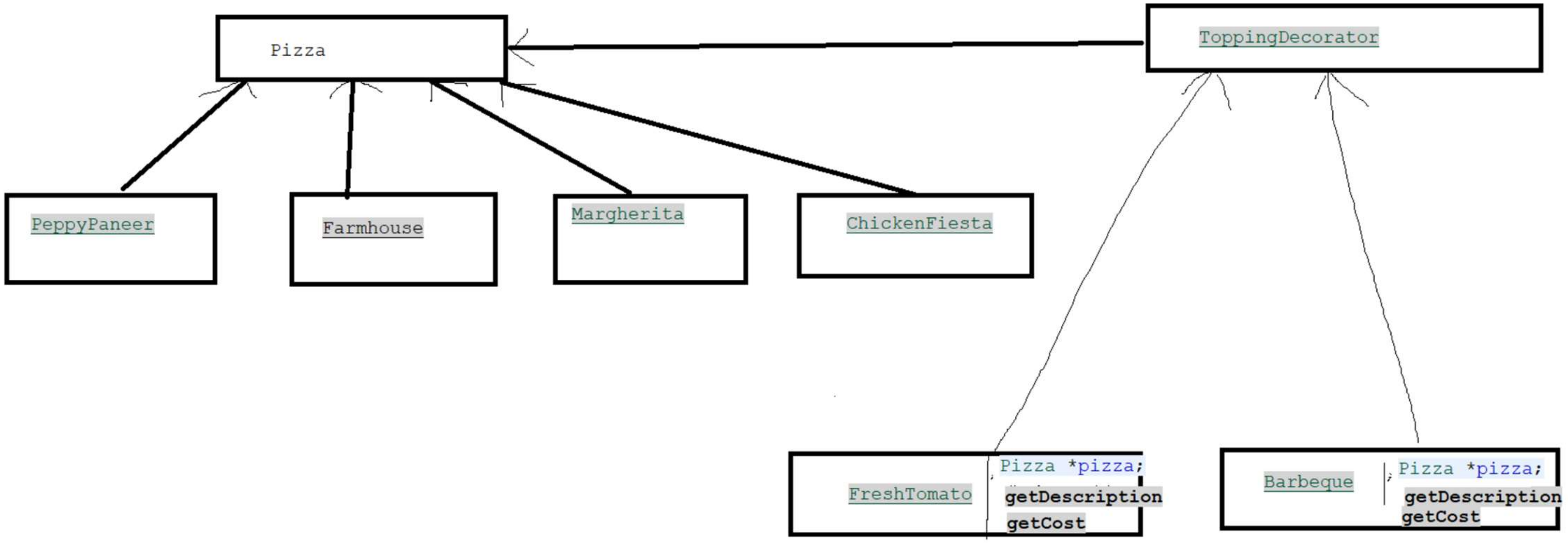
DECORATOR PROPERTIES

- Decorators have the same super type as the object they decorate.
- You can use multiple decorators to wrap an object.
- Since decorators have same type as object, we can pass around decorated object instead of original.
- We can decorate objects at runtime.

DEFINATION

- The decorator pattern attaches additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.




Pizza
ToppingDecorator
PeppyPaneer
Farmhouse
Margherita
ChickenFiesta
FreshTomato
Pizza *pizza;
getDescription
getCost
Barbeque
Pizza *pizza;
getDescription
getCost