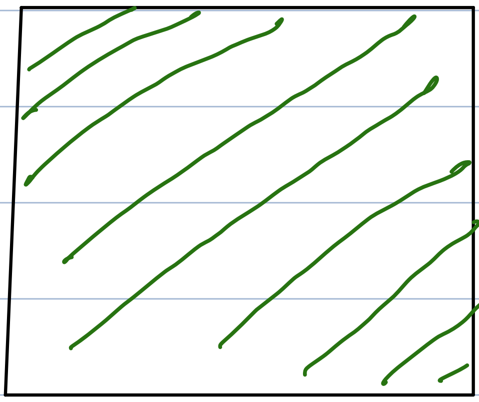
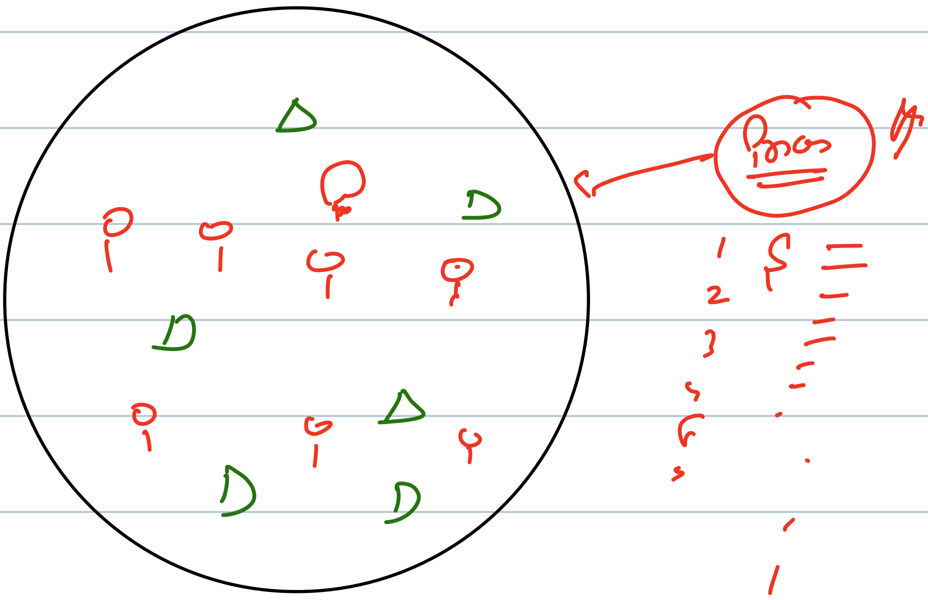
Agenda: - Decorator
- UML

☑ Gift wrap



Item + 30



① { Pizza : 300 }

⑪ Pizza : 300
Mush : 20
- 50

when ever we have customization & $\underbrace{\text{Price}}_{X}$ varies by customization.

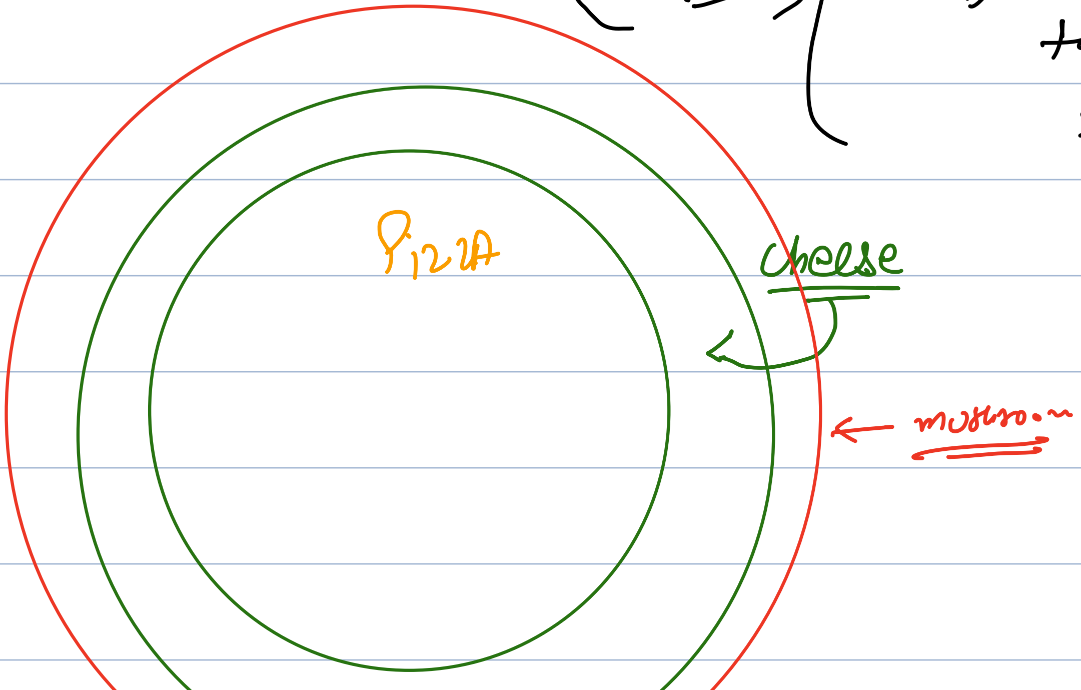
Product :

List<addOns...

get_Price_methods:

$$\left\{ \begin{array}{l} O(?) \\ \text{moustar} \\ \underline{\underline{\text{CPP}}} \end{array} \right\}$$

```
for a in order
  if a == m
    Price + =90
  if a == m
    +90
   :
```



```
ClassPizzA :                    class   PlainPizzA(PizzA):
    get_Price():
                                            get_Price :
    get_Bread():                                return 300;

                                class   Addons( ABC):
                                    def   __init__(PizzA):
                                        self.PizzA = PizzA

                      class   Cheese(Addons):

                                get_Price( ):
                                        self.PizzA.Price + 30

                        Class   mushroom:

                                  get_Price ( ):
                                          return   self.PizzA.get
                                                      Price + 40;
```

P_1 = PlainPizzA()

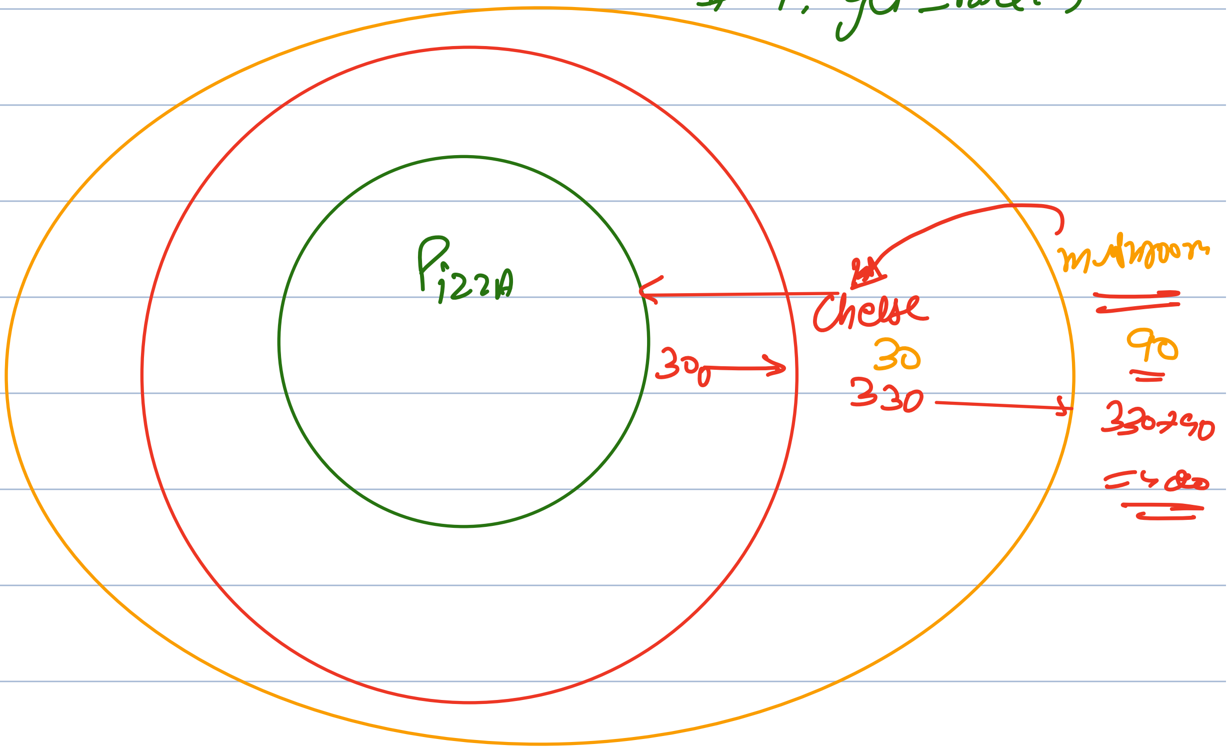
P. get_Price()   # 300

P_2 = Cheese(P_1)

P. get_Price()   # 330

P = mushroom(P_2)

P. get_Price()

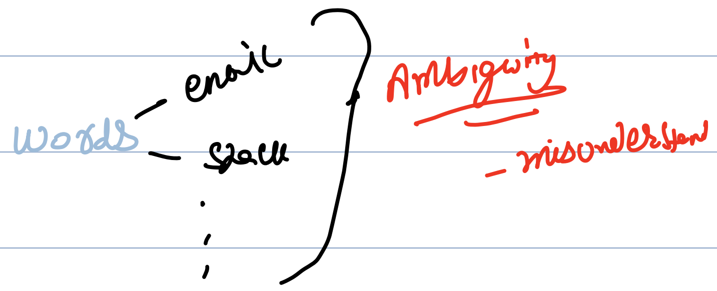


UML : Unified Modeling Lang

1. Client 2. Archi

3. eng. man 4. Product

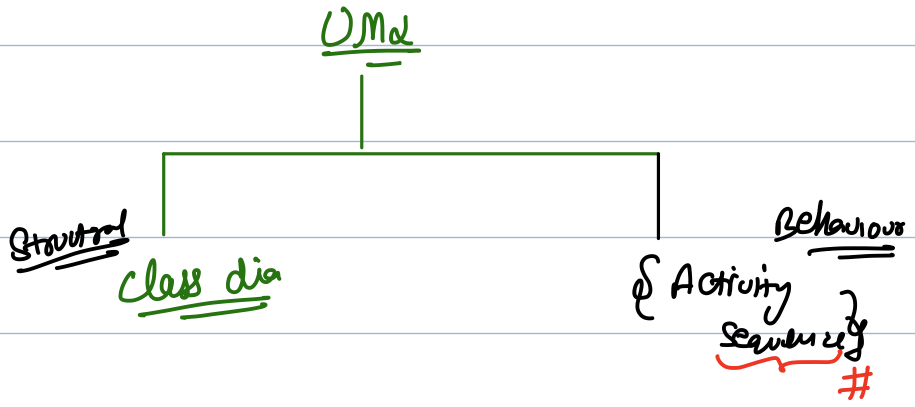
5. Business



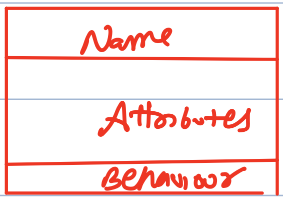
A Pictural way :

flow charts / dia /

Images



Class dia



Component: 1. Class
2 Interface
3. Abc class
4. enums

2. Show relation b/w all Classes:
1. who implement which Interface
2. which class extend which class

Attributes :

Access modifier name : datatype

Public +
Private -
Protected #

+ age : int

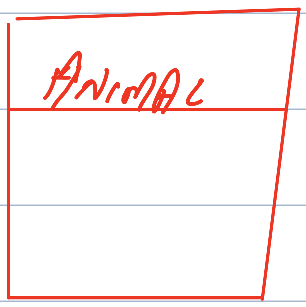
methods:

Access modifier method_name(dType): return dType

+ get_Price(PizzA) : int

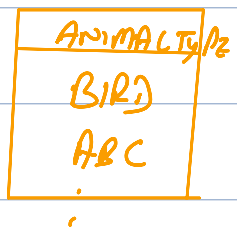
# How To write ABC:

ITALIC

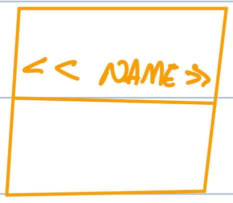


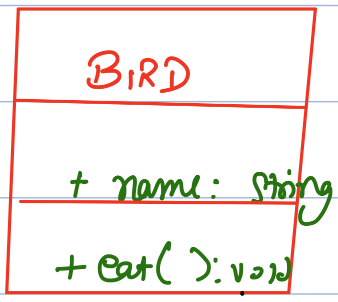
# Enums:

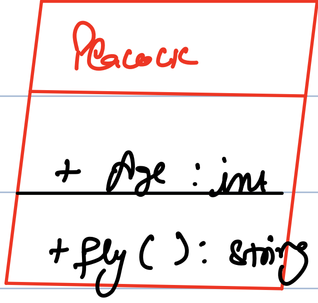
ALL CARS



# INTERFACE





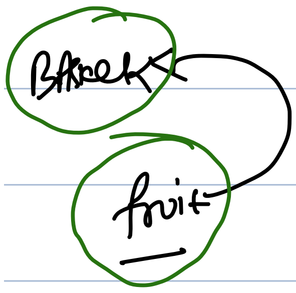


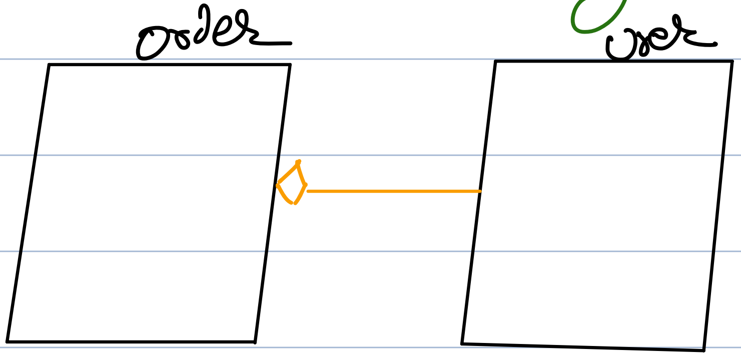
what are relationship of classes.

Association → HAS - A

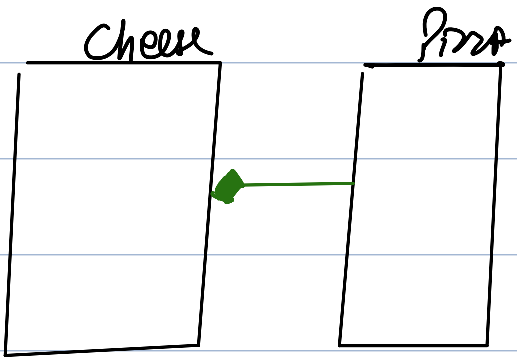
inheritence → IS - A

Aggrigation
- loose coupling
- Entity can exist indipendently
- collecting

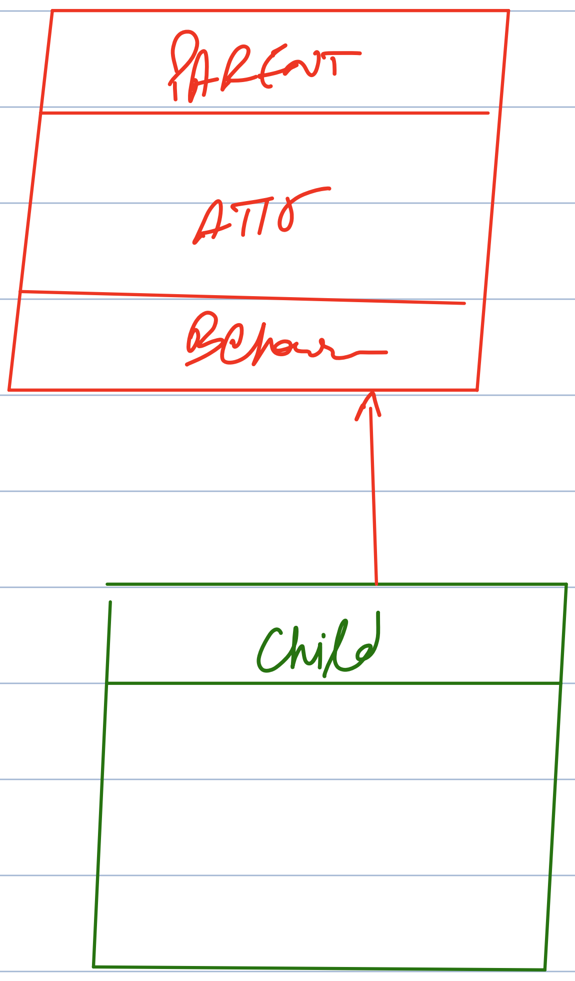


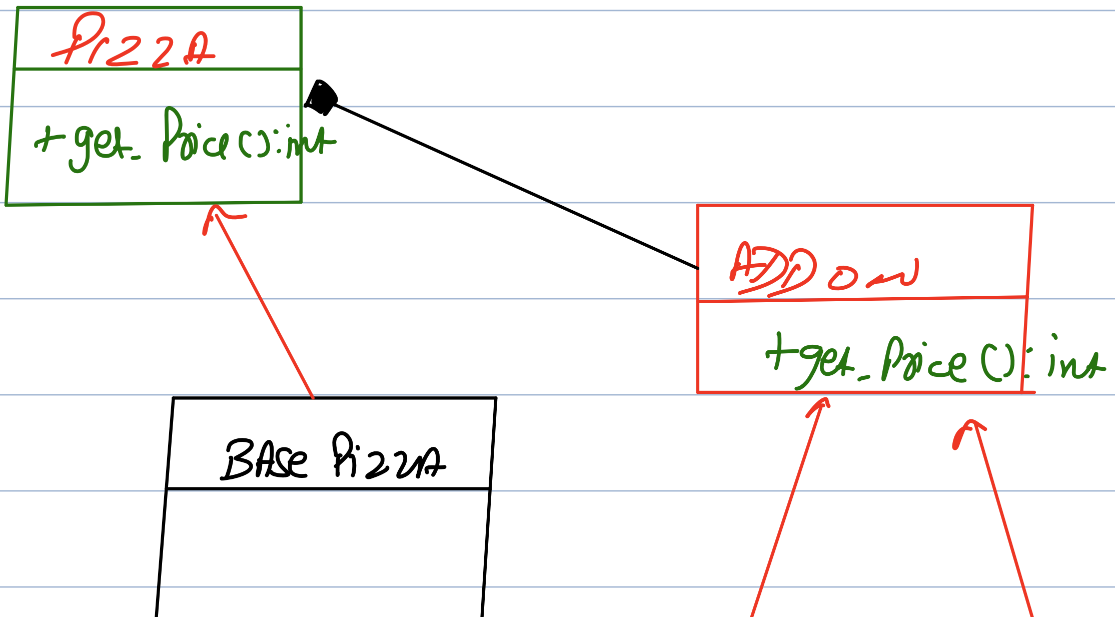


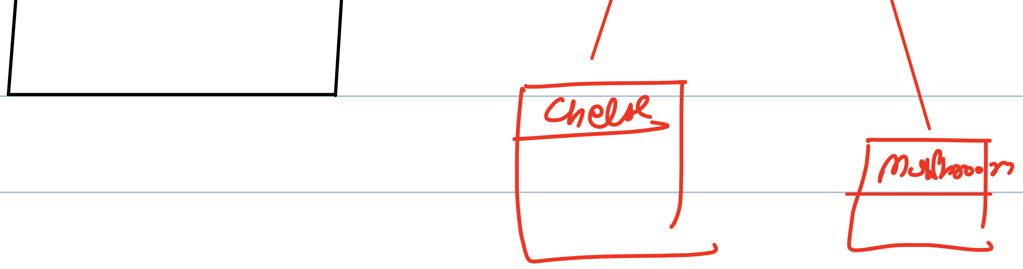
composition
- Tight
- if A exist then only B exist
- Creating. Cheese → pizza ?



INHERITANCE: IS-A






Cheese
Mushroom