Python Fundamentals day 57

Today's Agenda

- Has-A relationship

Has-A relationship

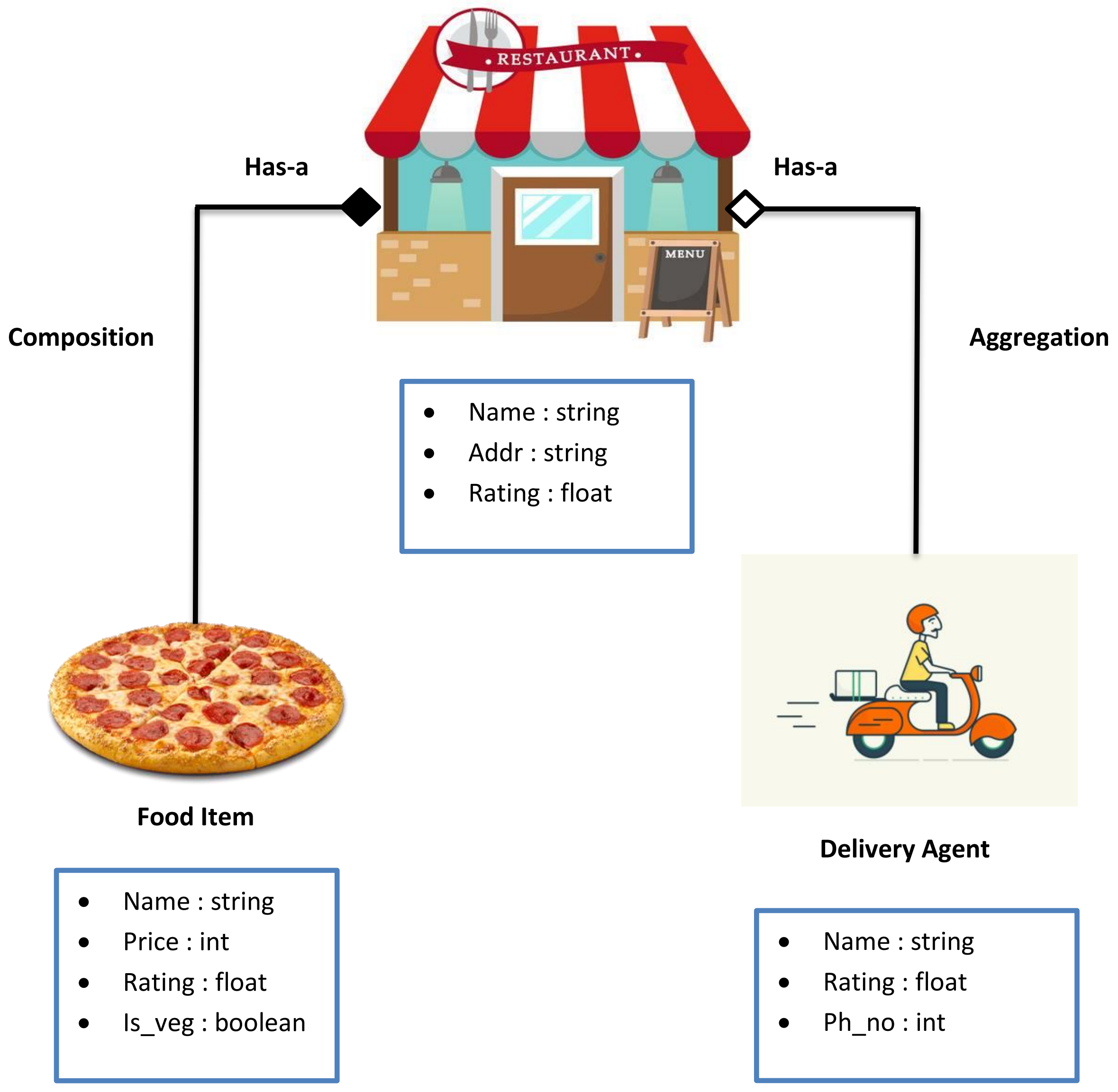
We have seen is-a relationship which can be achieved using inheritance but there is also has-a relationship which can be achieved using aggregation and composition.

Composition and aggregation are specialised form of Association. Where Association is a relationship between two classes without any rules.

In composition, one of the classes is composed of one or more instance of other classes. In other words, one class is container and other class is content and if you delete the container object then all of its contents objects are also deleted.

Aggregation is a weak form of composition. If you delete the container object contents objects can live without container object.

Let us take an example of restaurant and understand it correctly



- Name : string
- Addr : string
- Rating : float

Food Item

- Name : string
- Price : int
- Rating : float
- Is_veg : boolean

Delivery Agent

- Name : string
- Rating : float
- Ph_no : int

Let us write the code for above UML

```
class FoodItem:

    def __init__(self,name,price,rating,is_veg):
        self.name=name
        self.price=price
        self.rating=rating
        self.is_veg=is_veg

class DeliverAgent():

    def __init__(self,name,rating,ph_no):
        self.name=name
        self.rating=rating
        self.ph_no=ph_no

class Restaurant:

    def __init__(self,name,addr,rating):
        self.name=name
        self.addr=addr
        self.rating=rating
        self.pizza = FoodItem('Pizza',500,4.5,False)

    def assign_delivery_agent(self,agent):
        self.agent=agent

def main():
    r=Restaurant('XYZ','Bangalore',4.7)
    steve=DeliverAgent('Steve',4.2,9900886655)
    r.assign_delivery_agent(steve)
    print(r.pizza.price)
    print(r.agent.name)

main()
```

Output:

```
In [2]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
500
Steve
```

Let us see if we can access food item and delivery agent without the restaurant

```python
class FoodItem:

    def __init__(self,name,price,rating,is_veg):
        self.name=name
        self.price=price
        self.rating=rating
        self.is_veg=is_veg

class DeliverAgent():

    def __init__(self,name,rating,ph_no):
        self.name=name
        self.rating=rating
        self.ph_no=ph_no

class Restaurant:

    def __init__(self,name,addr,rating):
        self.name=name
        self.addr=addr
        self.rating=rating
        self.pizza = FoodItem('Pizza',500,4.5,False)

    def assign_delivery_agent(self,agent):
        self.agent=agent

def main():
    r=Restaurant('XYZ','Bangalore',4.7)
    steve=DeliverAgent('Steve',4.2,9900886655)
    r.assign_delivery_agent(steve)
    print(r.pizza.price)
    print(r.agent.name)
    #print(pizza.price) throw error(tight bound-composition)
    print(steve.name) # loose bound - aggregation

main()
```

Output:

```
In [4]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
500
Steve
Steve
```

Next let us see if we remove restaurant from the equation can we still access food and delivery agent

```
class FoodItem:

    def __init__(self,name,price,rating,is_veg):
        self.name=name
        self.price=price
        self.rating=rating
        self.is_veg=is_veg

class DeliverAgent():

    def __init__(self,name,rating,ph_no):
        self.name=name
        self.rating=rating
        self.ph_no=ph_no

class Restaurant:

    def __init__(self,name,addr,rating):
        self.name=name
        self.addr=addr
        self.rating=rating
        self.pizza = FoodItem('Pizza',500,4.5,False)

    def assign_delivery_agent(self,agent):
        self.agent=agent

def main():
    r=Restaurant('XYZ','Bangalore',4.7)
    steve=DeliverAgent('Steve',4.2,9900886655)
    r.assign_delivery_agent(steve)
    print(r.pizza.price)
    print(r.agent.name)

    del r
    print(r.pizza.price)
    print(r.agent.name)
main()
```

Output:

```
  File "C:/Users/rooman/OneDrive/Desktop/python/test.py",
line 35, in main
    print(r.pizza.price)

UnboundLocalError: local variable 'r' referenced before
assignment
```

We cannot access food without restaurant which makes complete sense but we should we able to access delivery agent without restaurant

```
class FoodItem:

    def __init__(self,name,price,rating,is_veg):
        self.name=name
        self.price=price
        self.rating=rating
        self.is_veg=is_veg

class DeliverAgent():

    def __init__(self,name,rating,ph_no):
        self.name=name
        self.rating=rating
        self.ph_no=ph_no

class Restaurant:

    def __init__(self,name,addr,rating):
        self.name=name
        self.addr=addr
        self.rating=rating
        self.pizza = FoodItem('Pizza',500,4.5,False)

    def assign_delivery_agent(self,agent):
        self.agent=agent

def main():
    r=Restaurant('XYZ','Bangalore',4.7)
    steve=DeliverAgent('Steve',4.2,9900886655)
    r.assign_delivery_agent(steve)
    print(r.pizza.price)
    print(r.agent.name)

    del r
    #print(r.pizza.price)
    print(steve.name)
main()
```

Output:

```
In [6]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
500
Steve
Steve
```

We see that Food item is outside the restaurant class, which means by creating an object of food item the instances can still be accessed. But that is not how things are in reality. Food items are present only inside restaurant. Let us see if can implement the same even in the above code.

```
class DeliverAgent():

    def __init__(self,name,rating,ph_no):
        self.name=name
        self.rating=rating
        self.ph_no=ph_no

class Restaurant:

    class FoodItem: #innerclass

        def __init__(self,name,price,rating,is_veg):
            self.name=name
            self.price=price
            self.rating=rating
            self.is_veg=is_veg

    def __init__(self,name,addr,rating):
        self.name=name
        self.addr=addr
        self.rating=rating
        self.pizza = Restaurant.FoodItem('Pizza',500,4.5,False)

    def assign_delivery_agent(self,agent):
        self.agent=agent

def main():
    r=Restaurant('XYZ','Bangalore',4.7)
    steve=DeliverAgent('Steve',4.2,9900886655)
    r.assign_delivery_agent(steve)
    print(r.pizza.price)
    print(r.agent.name)

main()
```

Output:

```
In [7]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
500
Steve
```

Make sure whenever you are trying to access fooditem, first call restaurant.