Class reusability
Object oriented application is a collection of classes. The content of one class can be used inside another class in different ways.

1. Composition (Has-A)
2. Aggregation (Use-A)
3. Inheritance (IS-A)

Composition (Has-A)
Composition is a process of creating an object of one class inside another class (OR) a class refers to the members of another class.

In composition the life time contained object is until container object is exists. If container object is destroyed, contained objects also destroyed.

Example: Book has Pages
Car has Engine
Person has Address

Example:

```
class Engine:
    def start(self):
        print("Engine Start...")
    def stop(self):
        print("Engine Stop...")


class Car:

    def __init__(self):
        self.e=Engine()
    def start(self):
        self.e.start()
    def stop(self):
        self.e.stop()

audi=Car()
audi.start()
audi.stop()
```

Output

```
Engine Start...
Engine Stop...
```

Example:

```
class Address:
    def __init__(self):
        self.__street=None
        self.__city=None
    def read_address(self):
        self.__street=input("Enter Street :")
        self.__city=input("Enter City :")
    def print_address(self):
        print(f'Street {self.__street}')
        print(f'City {self.__city}')

class Person:
    def __init__(self):
        self.__name=None
        self.__add1=Address()
        self.__add2=Address()
    def read_person(self):
        self.__name=input("Enter Name :")
        self.__add1.read_address()
        self.__add2.read_address()
    def print_person(self):
        print(f'Name {self.__name}')
        self.__add1.print_address()
        self.__add2.print_address()


p1=Person()
p1.read_person()
p1.print_person()
```

Output

```
Enter Name :naresh
Enter Street :ameerpet
```

```
Enter City :hyd
Enter Street :s.r.nager
Enter City :hyd
Name naresh
Street ameerpet
City hyd
Street s.r.nager
City hyd
```

Example:

```
class Battery:
    def charging(self):
        print("Charging....")


class Mobile:
    def __init__(self):
        self.__b=Battery()
    def plugin(self):
        self.__b.charging()


iphone=Mobile()
iphone.plugin()
```

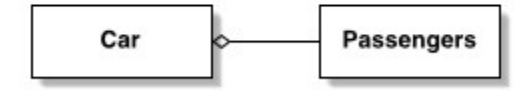
Output

```
Charging....
```

Aggregation

In aggregation contained object is not created inside contained class. Contained class object is created outside the container class and inject to container using constructor or method.

Composition: every car has an engine.



Aggregation: cars may have passengers, they come and go

Example:

```
class Course:
    def __init__(self,cn):
        self.__cname=cn
    def get_cname(self):
        return self.__cname

class Student:
    def __init__(self,n,c):
        self.__name=n
        self.__course=c
    def print_student(self):
        print(f'{self.__name}')
        print(f'{self.__course.get_cname()}')


course1=Course("Python")
stud1=Student("naresh",course1)
stud2=Student("suresh",course1)

stud1.print_student()
stud2.print_student()
```

Output

```
naresh
Python
suresh
Python
```