properties class
class property(fget=None, fset=None, fdel=None, doc=None)
Return a property attribute.
fget is a function for getting an attribute value. fset is a function for setting an attribute value

```
class Point:
    def __init__(self):
        self.__x=None
        self.__y=None
    def set_x(self,x):
        self.__x=x
    def set_y(self,y):
        self.__y=y
    def get_x(self):
        return self.__x
    def get_y(self):
        return self.__y
    x=property(fset=set_x,fget=get_x)
    y=property(fset=set_y,fget=get_y)
def main():
    p1=Point()
    p1.x=100
    print(p1.x)
    p1.y=200
    print(p1.y)
main()
```

Output:
```
100
200
>>>
```

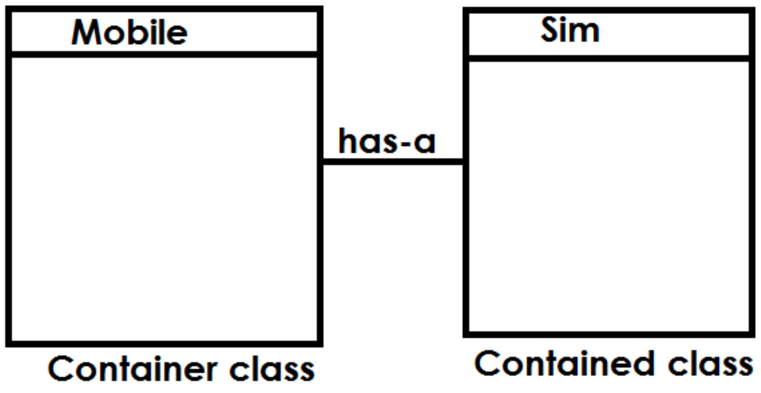
Class Reusability

Object oriented application is developed using multiple classes (OR) object oriented application is collection classes. The content of one class is used inside another class in different ways.

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

Composition (HAS-A)
Composition is process of creating an object of one class inside another class (OR) creating reference of one class inside another class is called composition.
Association between classes can be,

1. One-One
2. One-Many
3. Many-One
4. Many-Many



Example:

```
# contained class
class Engine:
    def start(self):
        print("Engine start...")
    def stop(self):
        print("Engine stop...")
#container class
class Car:
    def __init__(self):
        self.__e=Engine()
    def start(self):
        self.__e.start()
    def stop(self):
        self.__e.stop()
def main():
    car1=Car()
    car1.start()
    car1.stop()
main()
```

Output:

```
Engine start...
Engine stop...
>>>
```

Example:

#one-one

```
class Address:
    def __init__(self):
        self.__street=None
        self.__city=None
    def read_address(self):
        self.__street=input("Street:")
        self.__city=input("City:")
    def print_address(self):
        print(f'Street:{self.__street}')
        print(f'City:{self.__city}')
class Person:
    def __init__(self):
        self.__name=None
        self.__add=Address()
    def read_person(self):
        self.__name=input("Name:")
        self.__add.read_address()
    def print_person(self):
        print(f'Name:{self.__name}')
        self.__add.print_address()
def main():
    p1=Person()
    p1.read_person()
    p1.print_person()
main()
```

Output

```
Name:naresh
Street:s.r.nager
City:hyd
Name:naresh
Street:s.r.nager
City:hyd
>>>
```

#one-many

```
class Course:
    def __init__(self):
        self.__cname=None
        self.__fees=None
    def set_cname(self,cname):
        self.__cname=cname
    def get_cname(self):
        return self.__cname
    def set_fees(self,fees):
        self.__fees=fees
    def get_fees(self):
        return self.__fees
    cname=property(fset=set_cname,fget=get_cname)
    fees=property(fset=set_fees,fget=get_fees)
class Student:
    def __init__(self):
        self.__name=None
        self.__courses={'c1':Course(),'c2':Course()}
    def read_student(self):
        self.__name=input("Name :")
        self.__courses['c1'].cname=input("CourseName1:")
        self.__courses['c1'].fees=input("Fees:")
        self.__courses['c2'].cname=input("CourseName2:")
        self.__courses['c2'].fees=input("Fees:")
    def print_student(self):
        print(f'Name :{self.__name}')
        for c in self.__courses:
            course=self.__courses[c]
            print(f'CourseName: {course.cname}\tFees:{course.fees}')

def main():
    stud1=Student()
    stud1.read_student()
    stud1.print_student()
main()
```

Output

```
Name :naresh
CourseName1:python
Fees:4000
CourseName2:java
Fees:2000
Name :naresh
CourseName: python   Fees:4000
CourseName: java     Fees:2000
>>>
```

Example:

```
#many-one
class Printer:
    def print_data(self,data):
        print(data)

class Notepad:
    p=Printer()
    def __init__(self):
        self.__text=None
    def set_text(self,text):
        self.__text=text
    def print_text(self):
        Notepad.p.print_data(self.__text)

def main():
    npad1=Notepad()
    npad2=Notepad()
    npad1.set_text("This text is belongs to notepad1")
    npad2.set_text("This text is belongs to notepad2")
    npad1.print_text()
    npad2.print_text()
main()
```

Output:

```
This text is belongs to notepad1
This text is belongs to notepad2
>>>
```