super() method

- **Understanding Python super() with __init__() methods**
- Python has a reserved method called “__init__.” In Object-Oriented Programming, it is referred to as a constructor. When this method is called it allows the class to initialize the attributes of the class. In an inherited subclass, a parent class can be referred with the use of the super() function. The super function returns a temporary object of the superclass that allows access to all of its methods to its child class.

Super function in single inheritance

- Let’s take the example of animals. Dogs, cats, and cows are part of animals. They also share common characteristics like –
- They are mammals.
- They have a tail and four legs.
- They are domestic animals.
- So, the classes dogs, cats, and horses are a subclass of animal class. This is an example of single inheritance because many subclasses is inherited from a single parent class.

```
# Python program to demonstrate
# super function

class Animals:
        # Initializing constructor
        def __init__(self):
                self.legs = 4
                self.domestic = True
                self.tail = True
                self.mammals = True

        def isMammal(self):
                if self.mammals:
                        print("It is a mammal.")

        def isDomestic(self):
                if self.domestic:
                        print("It is a domestic animal.")
```

```
class Dogs(Animals):
        def __init__(self):
                super().__init__()

        def isMammal(self):
                super().isMammal()

class Horses(Animals):
        def __init__(self):
                super().__init__()

        def hasTailandLegs(self):
                if self.tail and self.legs == 4:
                        print("Has legs and tail")

# Driver code
Tom = Dogs()
Tom.isMammal()
Bruno = Horses()
Bruno.hasTailandLegs()
```

Output:

```
It is a mammal.
Has legs and tail
```

Super function in multiple inheritances

- Let’s take another **example of a super function**, Suppose a class **canfly** and **canswim** inherit from a mammal class and these classes are inherited by the animal class. So the animal class inherits from the multiple base classes. Let’s see the use of **Python super with arguments** in this case

```
class Mammal():

        def __init__(self, name):
                print(name, "Is a mammal")

class canFly(Mammal):

        def __init__(self, canFly_name):
                print(canFly_name, "cannot fly")

                # Calling Parent class
                # Constructor
                super().__init__(canFly_name)

class canSwim(Mammal):

        def __init__(self, canSwim_name):

                print(canSwim_name, "cannot swim")

                super().__init__(canSwim_name)

class Animal(canFly, canSwim):

        def __init__(self, name):
                super().__init__(name)


# Driver Code
Carol = Animal("Dog")
```

Output:

The class Animal inherits from two-parent classes – canFly and canSwim. So, the subclass instance Carol can access both of the parent class constructors.

```
Dog cannot fly
Dog cannot swim
Dog Is a mammal
```

example:Python super()

```
class Emp():
        def __init__(self, id, name, Add):
                self.id = id
                self.name = name
                self.Add = Add

# Class freelancer inherits EMP
class Freelance(Emp):
        def __init__(self, id, name, Add, Emails):
                super().__init__(id, name, Add)
                self.Emails = Emails

Emp_1 = Freelance(103, "Suraj kr gupta", "Noida" , "KKK@gmails")
print('The ID is:', Emp_1.id)
print('The Name is:', Emp_1.name)
print('The Address is:', Emp_1.Add)
print('The Emails is:', Emp_1.Emails)
```

```python
class Person:
    def __init__(self, fname, lname, address,age):
        self.fname = fname
        self.lname = lname
        self.address = address
        self.age=age
    def display(self):
        print("First Name: ", self.fname)
        print("Last Name: ", self.lname)
        print("Address: ", self.address)
        print("Age: ", self.age)
class employee(Person):
    company="Microsoft"
    def __init__(self, fname, lname, address,age,salary,job_title):
        Person.__init__(self,fname, lname, address,age)
        self.salary = salary
        self.job_title=job_title
    def display1(self):
        super().display()
        print("salary: ", self.salary)
        print("job_title: ", self.job_title)
```

```
# person object
per = Person("Adam", "Ho", "1234 abc blvd",26)
per.display()
print("=========================================")
std = employee("Peter", "kee", "9876 xyz blvd", 28, 354684,"eng")
std.display1()
```

```
========== RESTART: C:\Python
First Name:  Adam
Last Name:  Ho
Address:  1234 abc blvd
Age:  26
=========================================
First Name:  Peter
Last Name:  kee
Address:  9876 xyz blvd
Age:  28
salary:  354684
job_title:  eng
```

Write python program for the UML diagram below using super function