In [3]:
```
#SingleInheritance
class Parent:
    def display(self):
        print("This is a parent class")
class Child(Parent):
    def show(self):
        print("This is a Child class")
obj=Child()
obj.display()#inherited
obj.show();#own method
```

```
This is a parent class
This is a Child class
```

In [9]:
```
class Animal:
    def sound(self):
        print("Animal having sound")
class Dog(Animal):#child
    def Barks(self):
        print("Dog barks")
d=Dog()
d.sound()
d.Barks()
```

```
Animal having sound
Dog barks
```

In [11]:
```
#Multilevel
class Employee:
    def __init__(self,name,emp_id):
        self.name=name
        self.emp_id=emp_id
    def show_Employee_info(self):
        print(f"Employee Name:{ self.name},ID:{self.emp_id}")
class Developer(Employee):
     def __init__(self,name,emp_id,language):
        super().__init__(name,emp_id)
        self.language=language
     def show_developer_info(self):
         print(f"Developer Language :{self.language}")
class TechLead(Developer):
    def __init__(self,name,emp_id,language,team_size):
        super().__init__(name,emp_id,language)
        self.team_size=team_size
    def show_techlead_info(self):
         print(f"Tech lead of :{self.team_size} developers")

    def show_full_info(self):
        self.show_Employee_info()
        self.show_developer_info()
        self.show_techlead_info()
lead=TechLead("Suraj",101,"python",5)
lead.show_full_info()
```

```
Employee Name:Suraj,ID:101
Developer Language :python
Tech lead of :5 developers
```

In [21]:
```python
#heirarchical
class Vehicle:
    def start(self):
        print("Starting vehicle")
class Car(Vehicle):
    def drive(self):
        print("Driving the car")
class bike(Vehicle):
    def ride(self):
        print("riding Bike")
c=Car()
c.start()
c.drive()
b=bike()
b.ride()
b.start()
```

```
Starting vehicle
Driving the car
riding Bike
Starting vehicle
```

In [25]:
```python
#multiple Inheritance
class Father:
    def gardening(self):
        print("Loves Gardening")
class Mother:
    def cooking(sef):
        print("loves cooking")
class Child(Father,Mother):
    def play(self):
        print("Loves Playing")
c=Child()
c.gardening()
c.cooking()
c.play()
```

```
Loves Gardening
loves cooking
Loves Playing
```

In [45]:
```python
#Hybrid inheritance
#employee-Base class
#Manager-->inherit from Employee
#Developer-->inherit from Employee
#Tech_lead-->inherit from Manager and Developer(hybrid ingeritanve

class Employee:
    def __init__(self,name,emp_id):
        self.name=name
        self.emp_id=emp_id
```

```
    def show_Employee_info(self):
        print(f"Employee Name:{ self.name},ID:{self.emp_id}")
class Manager(Employee):
    def __init__(self,name,emp_id,Department):
        super().__init__(name,emp_id)
        self.Department=Department
    def show_manger(self):
        print(f"Department of Manager:{ self.Department}")


class Developer(Employee):
    def __init__(self,name,emp_id,language):
        super().__init__(name,emp_id)
        self.language=language
    def show_developer_info(self):
        print(f"he specialized in :{self.language}")
class TechLead(Manager,Developer):
    def __init__(self,name,emp_id,Department,language,team_size):
        Manager.__init__(self,name,emp_id,Department)
        Developer.__init__(self,name,emp_id,language)
        self.team_size=team_size
    def show_techlead_info(self):
        print(f" Lead Teams of :{self.team_size} developers")

    def show_full_info(self):
        self.show_Employee_info()
        self.show_developer_info()
        self.show_manger()
        self.show_techlead_info()
lead=TechLead("Suraj",101,"IT ","Java",15)
lead.show_full_info()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[45], line 40
     38         self.show_manger()
     39         self.show_techlead_info()
---> 40 lead=TechLead("Suraj",101,"IT ","Java",15)
     41 lead.show_full_info()

Cell In[45], line 29, in TechLead.__init__(self, name, emp_id, Department, language,
team_size)
     28 def __init__(self,name,emp_id,Department,language,team_size):
---> 29     Manager.__init__(self,name,emp_id,Department)
     30     Developer.__init__(self,name,emp_id,language)
     31     self.team_size=team_size

Cell In[45], line 15, in Manager.__init__(self, name, emp_id, Department)
     14 def __init__(self,name,emp_id,Department):
---> 15     super().__init__(name,emp_id)
     16     self.Department=Department

TypeError: Developer.__init__() missing 1 required positional argument: 'language'
```

In [ ]: