Experiment No : 09

Aim : Write a python program to understand different types of Inheritance.

Description :

Inheritance

The mechanism of deriving a new class from an old one(existing class)

Such that the new class inherits all the members(variables and methods)

of old class is called inheritance or derivation

Types of inheritance

1) single inheritance(if a class is derived from one base class(parent class), it is called single inheritance.

example: father class (parent class), son class(child class).

(father class) is child of (object).

Syntax;-

```
class ParentClassName(object):
     member of Parent Class
class ChildClassName(ParentClassName):
 member of Child Class
```

Example:-

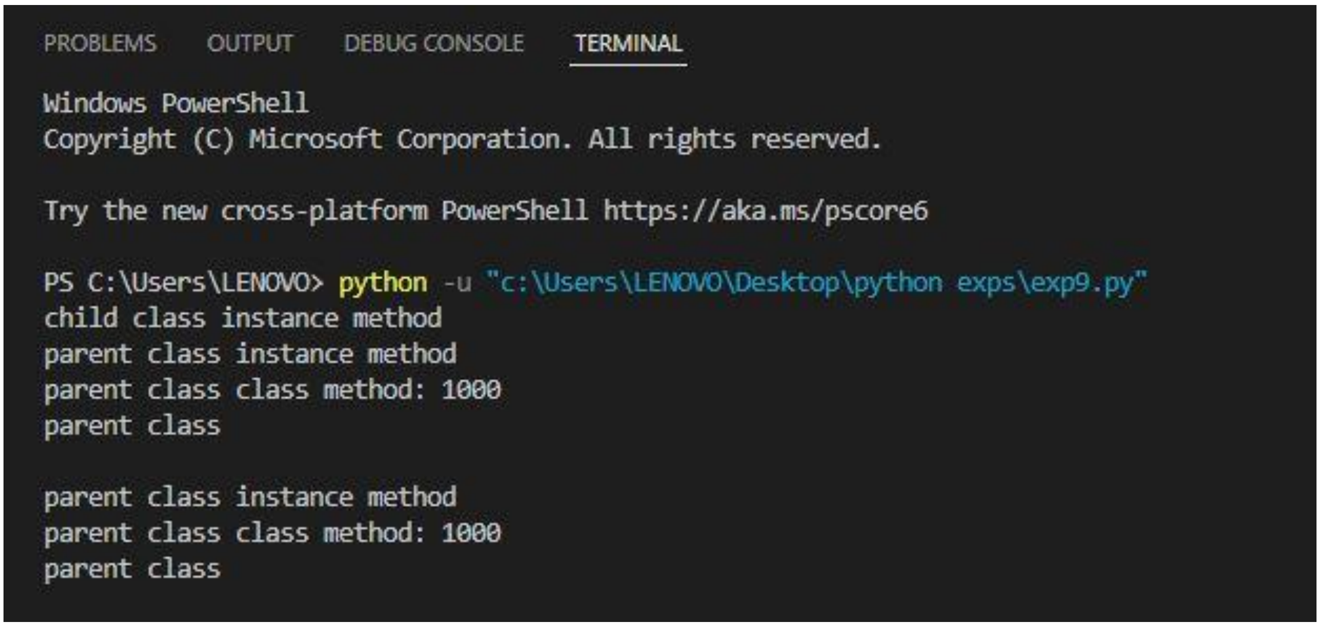
```
class Father:
     member of class Father
class son (Father):
     member of class Son
```

Implementation :

Code :

```python
class father:
       money = 1000
       def show(self):
           print("parent class instance method")
       @classmethod
       def showmoney(cls):
           print("parent class class method:", cls.money)
       @staticmethod
       def stat():
           a=10
           print("parent class")
class son(father):
       def disp(self):
           print("child class instance method")
s=son()
s.disp()
s.show()
s.showmoney()
s.stat()
print()
f=father()
f.show()
f.showmoney()
f.stat()
print()
#error
#f.disp()
```

Output :



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\LENOVO> python -u "c:\Users\LENOVO\Desktop\python exps\exp9.py"
child class instance method
parent class instance method
parent class class method: 1000
parent class

parent class instance method
parent class class method: 1000
parent class
```

2) Multi-level inheritance: In multi-level inheritance, the class inherits the feature of another derived class(child class)

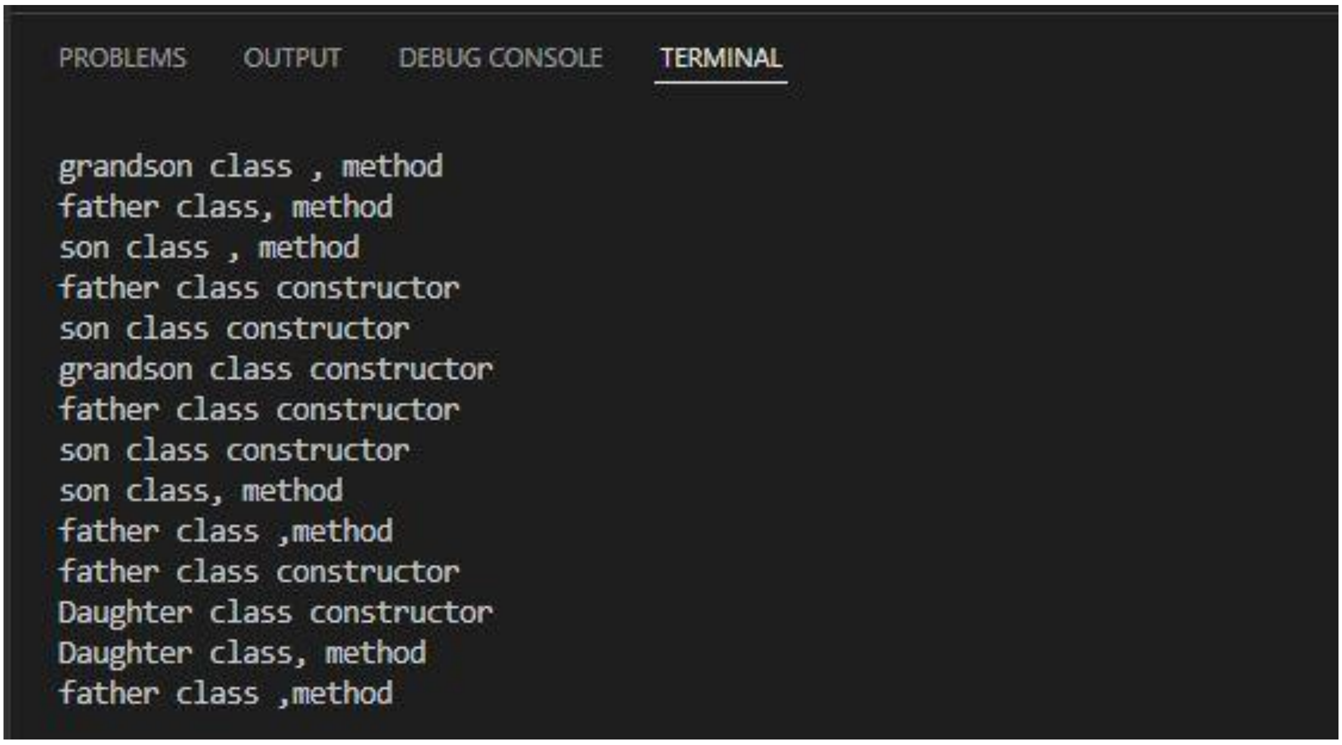
```
class ParentClassName(object):
    member of Parent Class


class ChildClassName(ParentClassName):
    member of Child Class
class GrandClassName(ChildClassName):
    member of Grand Child Class
# example:
class Father(object):
    member of class Father
class Son (Father):
    member of class Son
class GrandSon (Son):
    member of class GrandSon
```

Implementation :

Code :

```python
class father:

    def showF(self):
        print("father class, method")

class son(father):
    def showS(self):
        print("son class , method")
class grandson(son):
    def showG(self):
        print("grandson class , method")
g= grandson()
g.showG()
g.showF()
g.showS()


# CONSTRUCTOR
class father:
      def __init__(self):
           print("father class constructor")
      def showF(self):
           print("father class method")

class son(father):
      def __init__(self):
           super(). __init__()      #CALLING FATHER CLASS CONSTRUCTOR
           print("son class constructor")
      def showS(self):
           print("son class method")
class grandson(son):
      def __init__(self):
          super(). __init__()    # CALLING SON CLASS CONSTRUCTOR
          print("grandson class constructor")
      def showG(self):
          print("grandson class method")
g= grandson()
#g.showG()
#g.showF()
#g.showS()
```

Output :



```
grandson class , method
father class, method
son class , method
father class constructor
son class constructor
grandson class constructor
father class constructor
son class constructor
son class, method
father class ,method
father class constructor
Daughter class constructor
Daughter class, method
father class ,method
```

3) Hierarchical inheritance

syntax:

```
class ParentClassName(object):
    member of Parent Class
class ChildClassName1(ParentClassName):
    member of Child Class 2
class ChildClassName2(ParentClassName):
    member of Child Class 2
```

EXAMPLE:

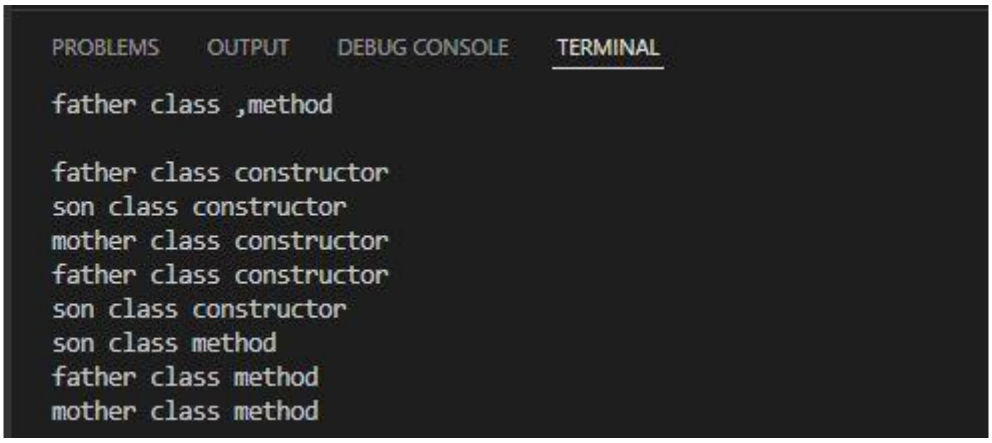
```
class Father(object):
    member of class Father
class Son (Father):
    member of class Son
class Daughter(Father):
    member of class Daughter
```

Implementation :

Code :

```
class father:
      def __init__(self):
         print("father class constructor")
      def showF(self):
         print("father class ,method")
class son(father):
      def __init__(self):
          super(). __init__()         # calling father class constructor
          print("son class constructor")
      def showS(self):
          print("son class, method")
class Daughter(father):
      def __init__(self):
          super(). __init__()     # calling father class constructor
          print("Daughter class constructor")
      def showD(self):
          print("Daughter class, method")

s=son()
s.showS()
s.showF()
#error
#s.showD()

d=Daughter()
d.showD()
d.showF()

print()
# son  and daughter will not call each others constructor
s=son()
```

output :



```
father class ,method

father class constructor
son class constructor
mother class constructor
father class constructor
son class constructor
son class method
father class method
mother class method
```

(4) Multiple Inheritance

if a class is derived from more than one parent class , then it is called multiple inheritance.

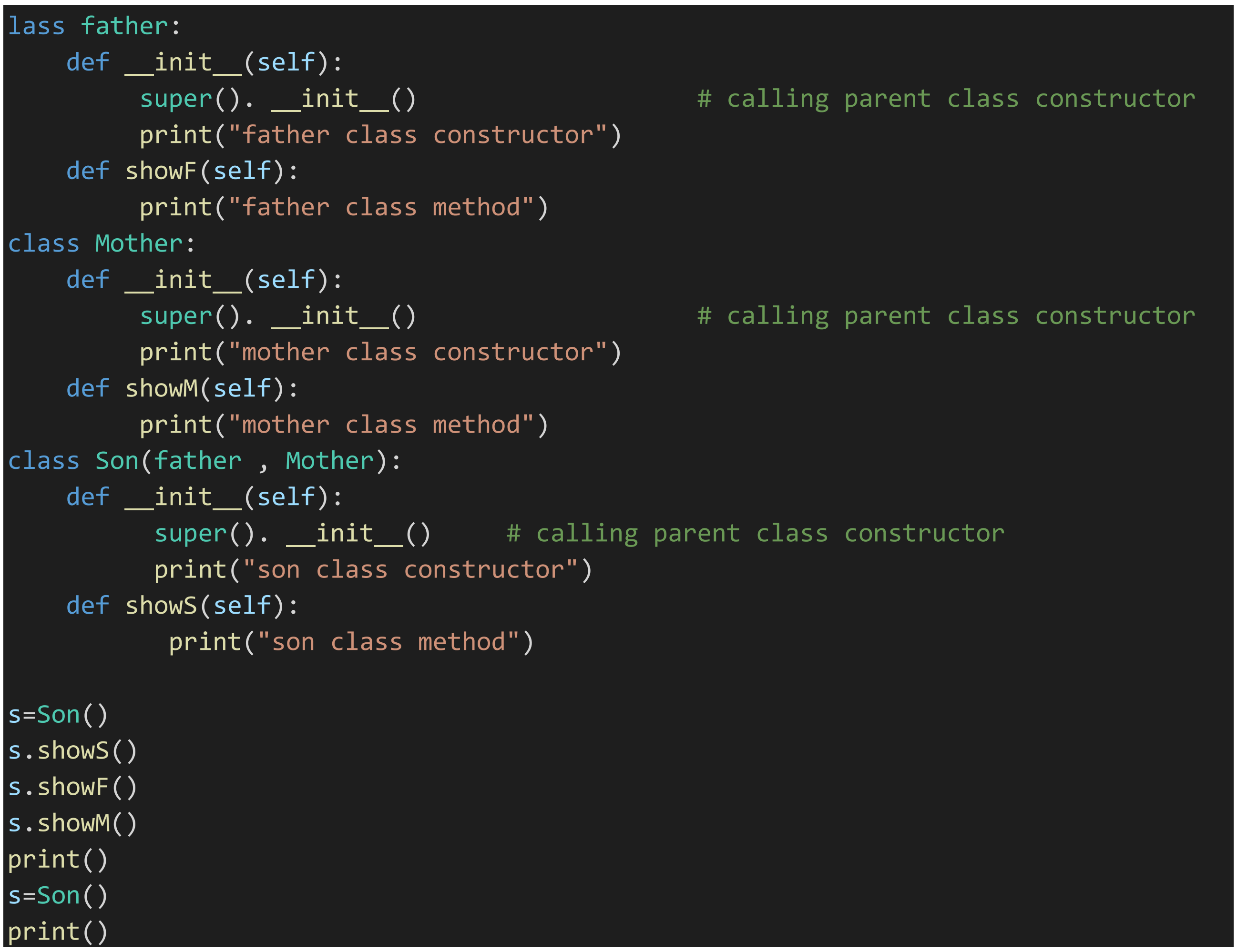
Syntax;-

```
class ParentClassName1(object):
    member of Parent Class
class ParentClassName2(object):
    member of Parent Class
class ChildClassName(ParentClassName1,ParentClassName2):
member of Child Class
```

```
# EXAMPLE:
class Father(object):
    member of class Father
class Mother(object):
    member of class Mother
class Son(Father , Mother):
    member of class Son
```

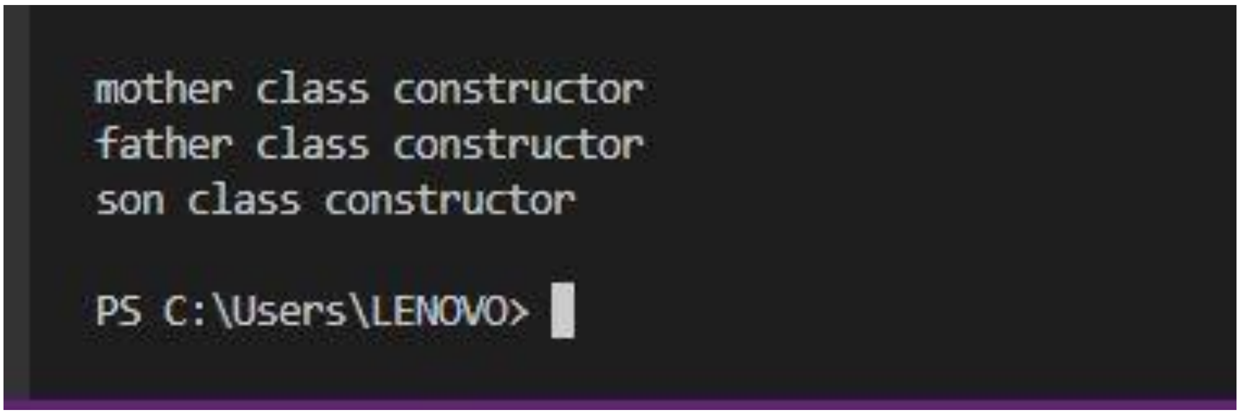
Implementation :

Code :



```
lass father:
    def __init__(self):
         super(). __init__()                  # calling parent class constructor
         print("father class constructor")
    def showF(self):
         print("father class method")
class Mother:
    def __init__(self):
         super(). __init__()                  # calling parent class constructor
         print("mother class constructor")
    def showM(self):
         print("mother class method")
class Son(father , Mother):
    def __init__(self):
          super(). __init__()     # calling parent class constructor
          print("son class constructor")
    def showS(self):
           print("son class method")

s=Son()
s.showS()
s.showF()
s.showM()
print()
s=Son()
print()
```

Output :



```
mother class constructor
father class constructor
son class constructor

PS C:\Users\LENOVO>
```

Conclusion : Therefore we have successfully implemented Inheritance and their types using python.