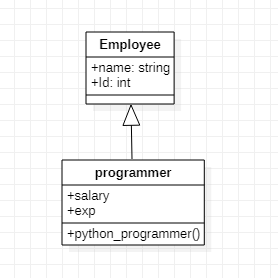
**Case Study 1**:

Create a class Employee and derive a child class programmer using public inheritance.

• Create class case diagram.

• Code the parent and derive classes on python



#case study 1

class Employee:

def \_\_init\_\_(self,name,id):

self.name=name

self.id=id

class Programmer(Employee):

def \_\_init\_\_(self,name,id):

super().\_\_init\_\_(name,id)

def Python\_programmer(self):

return(self.name,self.id)

p=Programmer("Huzaifa","021")

print(p.Python\_programmer())

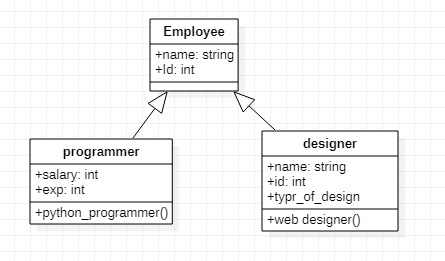
C:\Users\HASSAN ENTERPRISES\Desktop\oop lab 6 semester 2 pic 1.png

**Case Study 2**

Create a parent class Employee, derive child class Programmer and Designer using inheritance

• Create class case diagram.

• Code the parent and derive classes on python



#case study 2

class Employee:

def \_\_init\_\_(self,name,age):

self.name=name

self.age=age

def Detail(self):

return("Employee Data Name age and post is ",self.name,self.age)

class programmer:#yahan pr programmer ek dusarii class haii joo inherit haii upper walii class say likin

#eska oject ki help say may nay upper wali class ka attributes koo call kiyaaa

def \_\_init\_\_(self,salary,prolang,exp):

self.salary=salary

self.prolang=prolang

self.experience=exp

def Details(self):

return(self.salary,self.prolang,self.experience)

class Designer(Employee,programmer):

def \_\_init\_\_(self,name,age,salary,prolang,exp):

Employee.\_\_init\_\_(self,name,age)

programmer.\_\_init\_\_(self,salary,prolang,exp)

D=Designer("Muhammad",20,440000,"python","5-Year")

print(D.Details())

print(D.Detail())

C:\Users\HASSAN ENTERPRISES\Desktop\oop lab 6 semester 2 pic 2.png

**Case Study 3**:

Analyze the detail system of car loan management system. You can take help of internet to find you any car loan management system. Your task is as follows:

b. Define all the existing processes in loan management system.

c. Create classes.

d. Create use case diagram.

e. Make activity and sequence diagram.

f. Now select any class and try to enhance it according to your gap analysis.

g. Code the class for which you have enhance the process.

#case study 3

class Bank:

def \_\_init\_\_(self,bank\_location,bank\_Name,bank\_code,bank\_local\_currency,bank\_discount\_currency):

self.bank\_location=bank\_location

self.bank\_Name=bank\_Name

self.bank\_code=bank\_code

self.bank\_local\_currency=bank\_local\_currency

self.bank\_discount\_currency=bank\_discount\_currency

def Bank\_detailed(self):

return(f"The name of the Bank is {self.bank\_Name} near {self.bank\_location}. Bank code is {self.bank\_code} .Bank Local currency is {self.bank\_local\_currency} \n discount currency is {self.bank\_discount\_currency}")

class Account(Bank):

def \_\_init\_\_(self,bank\_location,bank\_Name,bank\_code,bank\_local\_currency,bank\_discount\_currency):

Bank.\_\_init\_\_(self,bank\_location,bank\_Name,bank\_code,bank\_local\_currency,bank\_discount\_currency)

class Banker(Account):

def \_\_init\_\_(self,bank\_location,bank\_Name,bank\_code,bank\_local\_currency,bank\_discount\_currency,Name):

Account.\_\_init\_\_(self,bank\_location,bank\_Name,bank\_code,bank\_local\_currency,bank\_discount\_currency)

self.Name=Name

def ManagerDetail(self):

return(f"Hello Sir my name is {self.Name}")

print("How can I help you sir? I deals All the customer those need loans if you can by a loan told me your account details")

def verification(self,customer\_detail,customer\_Name,customerAccount\_Number,x,loan):

self.x=x

self.customer\_detail=customer\_detail

self.customer\_Name=customer\_Name

self.customerAccount\_Number=customerAccount\_Number

self.loan=loan

if(self.x=="yes"):

print(f"The User name is {self.customer\_Name}\n The User Account number is {self.customerAccount\_Number}\n And Its Detail are {self.customer\_detail} .\n After Checking your previous record Bank give A loan of Rs={self.loan}")

else :

print("sir Kindy open your accounr First then I would give a loan")

x=input("Do you have an account Sir please reply in yes/no ")

customer\_Name=input("Please tell me your name Sir")

customerAccount\_Number=input("Please tell me your Account Number")

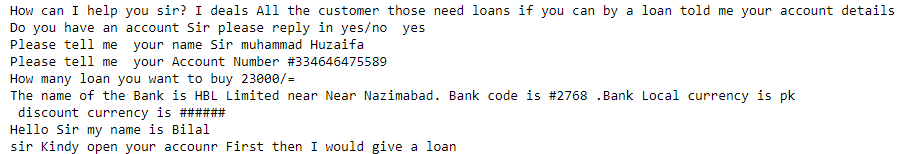
loan=input("How many loan you want to buy")

ba=Banker("Near Nazimabad","HBL Limited","#2768","pk","######","Bilal")

print(ba.Bank\_detailed())

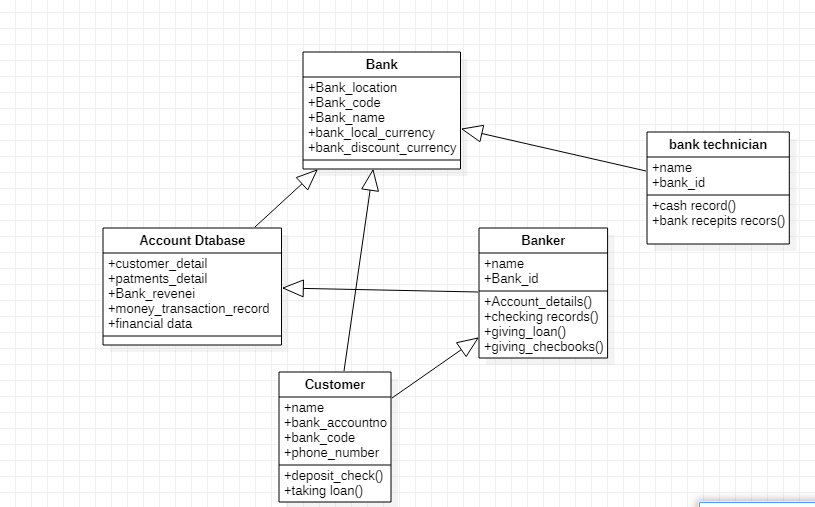
print(ba.ManagerDetail())

print(ba.verification(["BankBalance of your account is Rs=340000"],UserAccount\_Name,UserAccount\_Number,x,loan))

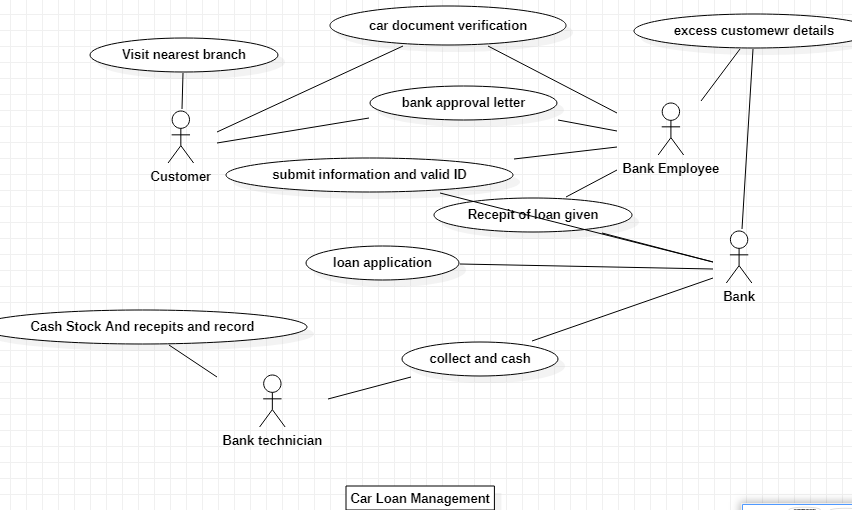


Car Loan Management System

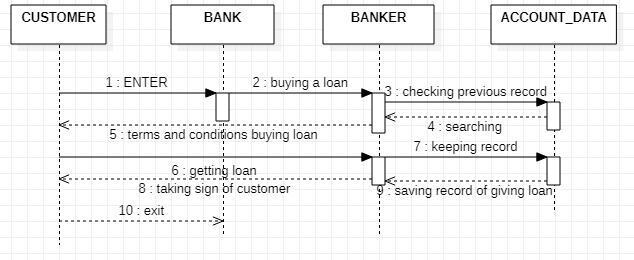
CLASS DIAGRAM:



USECASE DIAGRAM :



SEQUENCE DIAGRAM :



Analyze the detail system of Cargo Management System. You can take help of internet to find you any cargo management system. Your task is as follows:

a. Define all the existing processes in cargo management system.

b. Create the main classes.

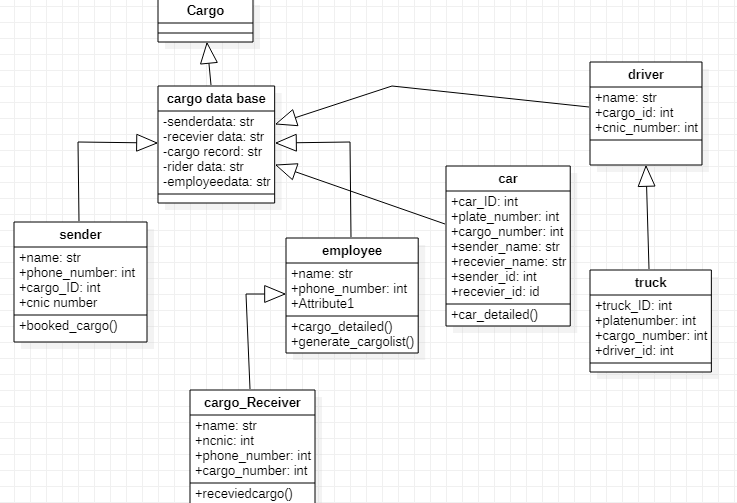
c. Create use case diagram.

d. Make activity and sequence diagram.

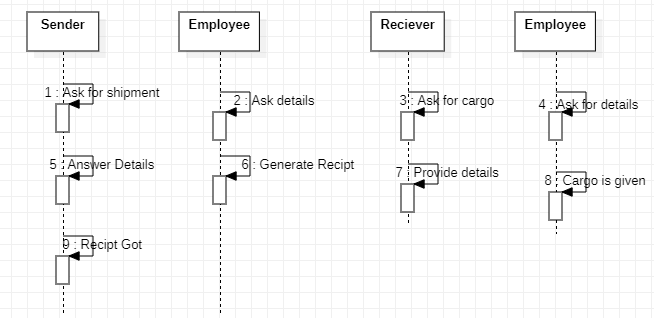
e. Now select any class and try to enhance it according to your gap analysis.

f. Code the class for which you have enhance the process.

CLASS DIAGRAM:



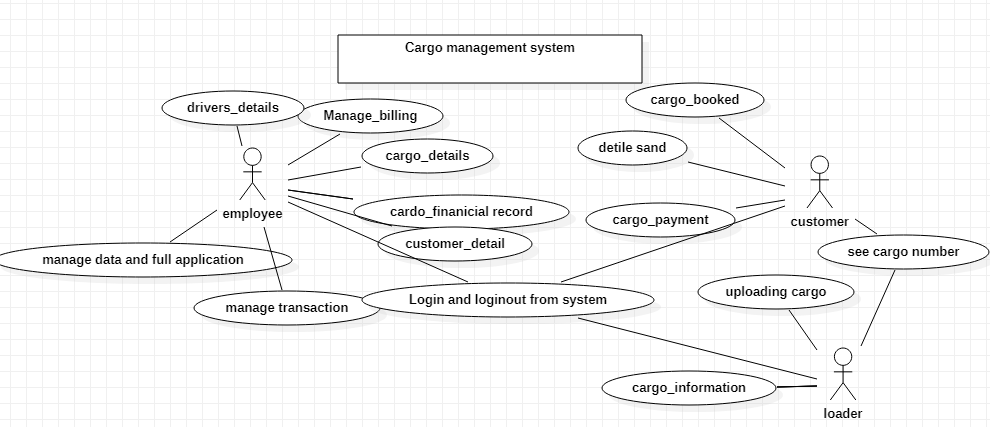
SEQUENCE DIAGRAM :



class cargo:

print("this cargomangement system")

USECASE DIAGRAM :



class cargo\_database(cargo):

def \_\_init\_\_(self,sender\_data,receiver\_data,cargo\_record,employee\_data,driver\_data):

self.sender\_data=sender\_data

self.receiver\_data=receiver\_data

self.cargo\_record=cargo\_record

self.employee\_data=employee\_data

self.driver\_data=driver\_data

def senderdetail(self):

print(self.sender\_data)

def receiverdetail(self):

print(self.receiver\_data)

def employeedetail(self):

print(self.employee\_data)

def driverdata(self):

print(self.driver\_data)

class sender(cargo\_database):

def \_\_init\_\_(self,sender\_data,receiver\_data,cargo\_record,employee\_data,driver\_data,name,phonenumber,cargo\_id,cnic\_number):

cargo\_database .\_\_init\_\_(self,sender\_data,receiver\_data,cargo\_record,employee\_data,driver\_data)

self.name=name

self.phonenumber=phonenumber

self.cargo\_id=cargo\_id

self.cnic\_number=cnic\_number

def booked\_cargo(self):

return('The name of sender is {} his cnic is {} and his phone no is {} and his cargo\_id is {}'.format(self.name,self.cnic\_number,self.phonenumber,self.cargo\_id))

class Employee(cargo\_database):

def \_\_init\_\_(self,sender\_data,receiver\_data,cargo\_record,employee\_data,driver\_data,name,phone\_number):

cargo\_database .\_\_init\_\_(self,sender\_data,receiver\_data,cargo\_record,employee\_data,driver\_data)

self.name=name

self.phone\_number=phone\_number

def cargo\_detailed(self):

return("The sendermail record are {} \n The recevierrecord are {}\n The cargorecord are {} and the driver data{}".format(self.sender\_data,self.receiver\_data,self.cargo\_record, self.driver\_data))

class cargo\_receiver(Employee):

def \_\_init\_\_(self,sender\_data,receiver\_data,cargo\_record,employee\_data,driver\_data,name,phone\_number,Receiver\_name,Receiver\_phone\_number,cnic\_number,cargo\_number):

cargo\_database .\_\_init\_\_(self,sender\_data,receiver\_data,cargo\_record,employee\_data,driver\_data)

self.Receiver\_name=Receiver\_name

self.Receiver\_phone\_number=Receiver\_phone\_number

self.cnic\_number=cnic\_number

self.cargo\_number=cargo\_number

def received\_cargo(self):

return(self.Receiver\_name,self.Receiver\_phone\_number,self.cnic\_number,self.cargo\_detailed)

receiver=cargo\_receiver(["Muhammadhuzaifa\n Muhamamd Bilal \n Muhammad safi"],["muhammad saad","safi uddin"],["cars"],["zubair","asad","faraz","shoaib"],"##","Ajaz","03360213796","sohail","03158287251","2353644745885959","56321")

sen=sender(["Muhammadhuzaifa\n Muhamamd Bilal \n Muhammad safi"],["muhammad saad","safi uddin"],["cars"],["zubair","asad","faraz","shoaib"],"##","Ajaz","03360213796","456789","74278723232")

print(receiver.senderdetail())

print(receiver.receiverdetail())

print(receiver.employeedetail())

print(receiver.driverdata())

print(sen.booked\_cargo())

print(receiver.cargo\_detailed())

print(receiver.received\_cargo())

