Student Name

Student ID

Submission Date

**Problem Analysis:**

In this structural project the major functionalities are within the sustained system. All the functional requirements are within how the data functionality is made possible. As an actual dependability feature, the relation therefore is actual and immediate to how the data itself is.

# **Classes, Attributes and Behaviors:**

class EmployeeDetails:

name

Department

ID

Title

Salary

class Employees(EmployeeDetails)

Name

ID

Department

Title

Salary

class Car:

Name

Price

title

class Sales:

UserID

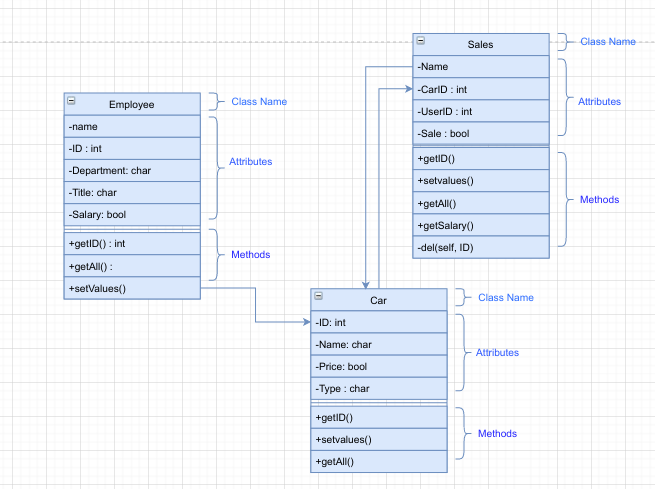
Name

CarID

Sale

The above are the major functional classes that are required within the program. Each design is managed by an actual feasible functionality that is within the data itself. All the rest of the functionality are actual and independent to the data flow.

**Class Design:**



**Pseudo code:**

class EmployeeDetails:

def \_\_init\_\_(self, name, id , department, title, salary):

self.name = name

self.id = id

self.department = department

self.title = title

self.salary = salary

def set(self):

pass

class Employees(EmployeeDetails):

def \_\_init\_\_(self,name = None, id = None , department = None, title = None, salary = None, db = None):

EmployeeDetails.\_\_init\_\_(self, name, id , department, title, salary)

self.conn= connect(db)

self.curr = self.conn.cursor()

def get\_id(self, id):

self.curr.execute()

return self.curr.fetchall()

def get\_all(self):

self.curr.execute("")

return self.curr.fetchall()

def set(self):

self.curr.execute()

self.conn.commit()

def delete(self, id):

sql = ''

self.curr.execute(sql, (id, ))

class Car:

def \_\_init\_\_(self, name= None, id = None, price = None, type = None, db = None):

self.name = name

self.id = id

self.price = price

self.type = type

self.conn= connect(db)

self.curr = self.conn.cursor()

def get\_id(self, id):

self.curr.execute()

return self.curr.fetchall()

def get\_all(self):

self.curr.execute()

return self.curr.fetchall()

def set(self):

self.curr.execute()

self.conn.commit()

def delete(self, id):

sql = ''

self.curr.execute(sql, (id, ))

class Sales:

def \_\_init\_\_(self, userid = None, name = None, carid= None, sale = None, db = None):

self.name = name

self.cid = carid

self.uid = userid

self.sale = sale

self.conn = connect(db)

self.curr = self.conn.cursor()

def set(self):

self.curr.execute()

self.conn.commit()

def get\_id(self, id):

self.curr.execute()

return self.curr.fetchall()

def get\_all(self):

self.curr.execute()

return self.curr.fetchall()

def delete(self, id):

sql = ''

self.curr.execute(sql, (id, ))

def get\_salary(self):

sales = self.curr.execute().fetchall()

manager = self.curr.execute().fetchall()

salesp = self.curr.execute().fetchall()

result = []

for i in salesp:

u = i

usdet = self.curr.execute().fetchall()

profit = 0

for j in sales:

if j[0] == i[0]:

carprice = self.curr.execute().fetchall()

profit += j[2] - carprice[0][0]

#print(usdet)

salary = (profit \* 6.5) + usdet[0][1]

mansal = (profit \* 3.5) +manager[0][1]

result.append((u[1], salary))

result.append((manager[0][0], mansal))

return result

def fill\_sale\_data():

salesdata = [["Joy Rogers", "ZX3",155000], ["Joy Rogers", "VX3", 57800],["Joy Rogers", "VX3", 55000], ["Joy Rogers","SX3", 89000], ["Joy Rogers","SX3", 93000],["Mark Jones", "VX3", 58000], ["Mark Jones","VX3", 58000], ["Mark Jones", "VX3", 158000],["Mark Jones", "VX3", 158000], ["Mark Jones", "VX3", 158000]]

cur = connect("").cursor()

for i in salesdata:

userid = cur.execute("Select ID\_number from Employee where Name = ?", (i[0], )).fetchall()

c= Sales(userid[0][0], i[0], i[1], i[2], db="")

c.set()

def fill\_emp\_data():

employeedata = [["Susan Meyers",47899, "Accounting","Manager", 37500], ["Mark Jones", 39119, "IT", "Salesperson", 26000], ["Joy Rogers", 81774, "Manufacturing", "Salesperson", 24000]]

for i in employeedata:

c = Employees(i[0], i[1], i[2], i[3], i[4], "")

c.set()

def fill\_car\_data():

car\_data=[["Jazz", "VX3", 55000, "Hatch"], ["Mark3", "SX3", 84000, "Sedan"], ["Wagoner", "ZX3", 125000, "SUV"]]

for i in car\_data:

c = Car(i[0], i[1], i[2], i[3], "")

c.set()

def print\_menu():

print("")

def few\_db\_fillers\_and\_actions():

print("\n\n")

fill\_emp\_data()#fill employee tabme

c = Employees(db = "")

#c.delete(81774)

print(c.get\_id(81774))

print("\n\n")

fill\_car\_data()#fill car table

c = Car(db = "")

print(c.get\_id("VX3"))

print("\n\n")

fill\_sale\_data()# fill sale table with data

c = Sales(db = "")

#print(c.get\_id(81774))

print(c.get\_salary())

#few\_db\_fillers\_and\_actions()

if \_\_name\_\_ == "\_\_main\_\_":

print\_menu()

ch = print("")

if ch == 1:

few\_db\_fillers\_and\_actions()

elif ch == 2:

c = Employees(db = "")

l = c.get\_all()

print("Name | ID Number | Department | Job | Basic Salary")

for i in l:

for c in i:

print(c, end=" ")

print()

**Conclusion:**

The data management of the above pseudo has an optimum structure that enables all the favorable features of the program to run. As an indicative suggestion all manipulation are constant towards how the program itself is able to run, recursively with a prospect towards how the data variable is initialized.