|  |  |  |
| --- | --- | --- |
| **Ex.No.9** | **Inheritance** | **Reg.No: URK23CS1261** |
| **27.3.24** |
| **9 A) Develop a python application using Inheritance concept to automate the salary calculation of**  **employee in an organization as per the salary band given below. Create a base class called Employee and derive sub classes as per the given table. Apply method overriding to implement the following services via menu driven interface.**  **a) Calculate Gross Salary**  **b) Calculate Net Salary**  **c) Calculate Tax**  **d) Print the Pay Details**  **Aim:** The objective of this program is to automate the salary calculation of employee in an organization as per the salary band  **Algorithm:**  Step 1: Start the program.  Step 2: Define a class named Employee with methods to calculate gross salary, tax, net salary, and display employee details.  Step 3: Define subclasses named Manager and Engineer inheriting from Employee with predefined salary components.  Step 4: Define a function named whom() to choose between Manager and Engineer.  Step 5: Enter an indefinite loop to display menu options and handle user input.  Step 6: Based on the selected option, call the corresponding function to calculate gross salary, net salary, tax, or display details using the whom() function.  Step 7: Print the calculated results or details.  Step 8: If an invalid option is selected, quit the program.  **Program:**  class Employee:  def CalculateGrossSalary(self):  gross\_salary = self.basicsalary + (self.basicsalary \* (self.dapay / 100)) + (self.basicsalary \* (self.hra / 100))  return gross\_salary  def CalculateTax(self):  return (self.CalculateGrossSalary() \* (self.tax / 100))  def CalculateNetSalary(self):  net\_salary = self.CalculateGrossSalary() - self.CalculateTax() - self.epf  return net\_salary  def display(self):  self.NetSalary = self.CalculateNetSalary()  return f"""  Basic Salary: {self.basicsalary}  DA Pay: {self.dapay}  HRA: {self.hra}  EPF: {self.epf}  Tax: {self.tax}  Net Salary: {self.NetSalary}  """  class Manager(Employee):  def \_\_init\_\_(self):  super().\_\_init\_\_()  self.basicsalary = 30000  self.dapay = 95  self.hra = 20  self.tax = 25  self.epf = 3000  class Engineer(Employee):  def \_\_init\_\_(self):  super().\_\_init\_\_()  self.basicsalary = 20000  self.dapay = 80  self.hra = 15  self.tax = 15  self.epf = 2000  def whom():  print('1.Manager\n2.Engineer')  option = int(input("Enter the choice: "))  if option == 1:  return Manager, 'Manager'  elif option == 2:  return Engineer, 'Engineer'  else:  print("Invalid Input")  quit()  while True:  print('Menu:\n1.Calculate Gross Salary\n2.Calculate Net Salary\n3.Calculate Tax\n4.Print the Pay Details')  option = int(input("Enter the option: "))  if option == 1:  obj, name = whom()  print(f"{name} --> Gross Salary: {obj().CalculateGrossSalary()}\n")  elif option == 2:  obj, name = whom()  print(f"{name} --> Net Salary: {obj().CalculateNetSalary()}\n")  elif option == 3:  obj, name = whom()  print(f"{name} --> Tax: {obj().CalculateTax()}\n")  elif option == 4:  obj, name = whom()  print(f"{name} --> Details: {obj().display()}\n")  else:  quit()  print("╔═════════════╗\n║   Tanvik    ║\n║ URK23CS1261 ║\n╚═════════════╝")  **Output:**        **Result:** Thus, The program has successfully produced the desired output. | | |
| **9 B) Develop a python application using Inheritance as per the following. Create a class Worker and**  **derive two classes DailyWorker and SalariedWorker from it. Every worker has name, salary rate.**  **Provide a method ComPay(int hours) to compute the week pay of every worker. A DailyWorker is**  **paid on the basis of number of days he/she works. The SalariedWorker gets paid the wage for 40 hours**  **a week no matter what actual hours is. Implement this scenario to calculate the pay of workers.**  **Aim:** The objective of this program is to develop menu-driven Python app managing payroll, comprising Worker, DailyWorker, and SalariedWorker classes, handling employee details efficiently.  **Algorithm:**  Step 1: Start the program.  Step 2: Define a class named Worker:  Step 2.1: Initialize the class with attributes name and salary\_rate.  Step 3: Define a subclass named DailyWorker inheriting from Worker:  Step 3.1: Define a method comp\_pay(days) to compute weekly pay based on days worked.  Step 4: Define a subclass named SalariedWorker inheriting from Worker:  Step 4.1: Define a method comp\_pay() to compute weekly pay for 40 hours worked.  Step 5: Enter an indefinite loop to display menu options and handle user input.  Step 6: Display menu options for Daily Worker and Salaried Worker.  Step 7: Take user input for the option.  Step 8: If the option is 1:  Step 8.1: Prompt user for Daily Worker details: name, salary rate, and days worked.  Step 8.2: Compute and print the weekly pay using the comp\_pay() method of DailyWorker.  Step 9: If the option is 2:  Step 9.1: Prompt user for Salaried Worker details: name and salary rate.  Step 9.2: Compute and print the weekly pay using the comp\_pay() method of SalariedWorker.  Step 10: If an invalid option is selected, quit the program.  Step 11: End the program.  **Program:**  class Worker:  def \_\_init\_\_(self, name, salary\_rate):  self.name = name  self.salary\_rate = salary\_rate  class DailyWorker(Worker):  def comp\_pay(self, days):  return self.salary\_rate \* days  class SalariedWorker(Worker):  def comp\_pay(self):  return self.salary\_rate \* 40    while True:  print('Menu:\n1.Daily Worker\n2.Salaried Worker')  option = int(input("Enter the option: "))  if option == 1:  print('Daily Worker')  name = input("Enter the name: ")  salar = float(input("Enter Salary Rate: "))  days = float(input("Enter days: "))  value = DailyWorker(name=name, salary\_rate=salar).comp\_pay(days)  print(f"Com Pay: {value}")  elif option == 2:  print('Daily Worker')  name = input("Enter the name: ")  salar = float(input("Enter Salary Rate: "))  value = SalariedWorker(name=name, salary\_rate=salar).comp\_pay()  print(f"Com Pay: {value}\n")  else:  quit()  print("╔═════════════╗\n║ Tanvik ║\n║ URK23CS1261 ║\n╚═════════════╝")  **Output:**    **Result:** Thus, The program has successfully produced the desired output. | | | |