

|  |  |
| --- | --- |
| NAME | Ali Mustafa shah |
| REGISTRATION NUMBER | SU92-BSDSM-S24-005 |
| SECTION | 2A |
| SEMESTER | 2ND |
| SUBJECT | OOP (LAB) |
| ASSIGNMENT | 11 |
| SUBMITTED TO | Sir Rasikh |

**LAB-TASK-11**

**Task 1;** **Program to manage employee personal details:**

**1. Define a parent class called Employee with private attributes name, age, and salary. Implement getter and setter methods for each attribute to ensure controlled access to the data.**

**2. Create a child class Manager inheriting from Employee. The Manager class should have an additional private attribute called department. Implement getter and setter methods for the department attribute.**

**3. Create another child class Worker inheriting from Employee. The Worker class should have an additional private attribute called hours\_worked. Implement getter and setter methods for the hours\_worked attribute.**

**4. Implement file handling to store and retrieve information about employees. Use a CSV file format to store the information in a structured manner, where each row represents an employee and each column represents an attribute (name, age, salary, department, hours\_worked). Ensure the information is stored and retrieved in a structured manner.**

**5. Develop functions to add new employees, display information of all employees, update employee information, and delete employees from the records. Ensure that these functions interact with the Employee class and its subclasses using appropriate encapsulation techniques.**

**6. Provide a user interface to interact with the program, allowing users to perform operations like adding, displaying, updating, and deleting employee information through a menu-driven interface. 7. Organize the code into multiple files Use import statements to manage dependencies between files.**

**Solution**

import csv

class Employee:

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

        self.\_\_name = name

        self.\_\_age = age

        self.\_\_salary = salary

    def get\_name(self):

        return self.\_\_name

    def set\_name(self, name):

        self.\_\_name = name

    def get\_age(self):

        return self.\_\_age

    def set\_age(self, age):

        self.\_\_age = age

    def get\_salary(self):

        return self.\_\_salary

    def set\_salary(self, salary):

        self.\_\_salary = salary

    def to\_csv(self):

        return f"{self.\_\_name},{self.\_\_age},{self.\_\_salary}"

class Manager(Employee):

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

        super().\_\_init\_\_(name, age, salary)

        self.\_\_department = department

    def get\_department(self):

        return self.\_\_department

    def set\_department(self, department):

        self.\_\_department = department

    def to\_csv(self):

        return f"{super().to\_csv()},{self.\_\_department}"

class Worker(Employee):

    def \_\_init\_\_(self, name, age, salary, hours\_worked):

        super().\_\_init\_\_(name, age, salary)

        self.\_\_hours\_worked = hours\_worked

    def get\_hours\_worked(self):

        return self.\_\_hours\_worked

    def set\_hours\_worked(self, hours\_worked):

        self.\_\_hours\_worked = hours\_worked

    def to\_csv(self):

        return f"{super().to\_csv()},{self.\_\_hours\_worked}"

class EmployeeManager:

    def \_\_init\_\_(self, filename='employees.csv'):

        self.filename = filename

        self.employees = []

    def load\_employees(self):

        try:

            with open(self.filename, mode='r') as file:

                reader = csv.reader(file)

                for row in reader:

                    if len(row) == 5:

                        emp = Manager(row[0], int(row[1]), float(row[2]), row[3])

                    elif len(row) == 4:

                        emp = Worker(row[0], int(row[1]), float(row[2]), int(row[3]))

                    self.employees.append(emp)

        except FileNotFoundError:

            print("File not found. No employees loaded.")

    def save\_employees(self):

        with open(self.filename, mode='w', newline='') as file:

            writer = csv.writer(file)

            for emp in self.employees:

                writer.writerow(emp.to\_csv().split(','))

    def add\_employee(self, emp):

        self.employees.append(emp)

        self.save\_employees()

    def display\_employees(self):

        if not self.employees:

            print("No employees found.")

            return

        for emp in self.employees:

            print(emp.to\_csv())

    def update\_employee(self, index, emp):

        if 0 <= index < len(self.employees):

            self.employees[index] = emp

            self.save\_employees()

    def delete\_employee(self, index):

        if 0 <= index < len(self.employees):

            del self.employees[index]

            self.save\_employees()

    def get\_employees(self):

        return self.employees

def display\_menu():

    print("\nEmployee Management System")

    print("1 Add the Employee")

    print("2 Display the Employee")

    print("3 Update the Employee details ")

    print("4 Delete the  Employee details ")

    print("5 Exit")

def main():

    emp\_manager = EmployeeManager()

    emp\_manager.load\_employees()

    while True:

        display\_menu()

        choice = input("Enter the number what your wan't : ")

        if choice == '1':

            emp\_type = input("Enter the emplyee place (worker or maniger) ").strip().lower()

            name = input("Enter name: ")

            try:

                age = int(input("Enter your age : "))

                salary = float(input("Enter the salary : "))

            except ValueError:

                print("Invalid input please enter the walid input : ")

                continue

            if emp\_type == 'maniger':

                department = input("Enter the department: ")

                emp = Manager(name, age, salary, department)

            elif emp\_type == 'worker':

                try:

                    hours\_worked = int(input("Enter the hour :  "))

                    emp = Worker(name, age, salary, hours\_worked)

                except ValueError:

                    print("Invalid input please enter")

                    continue

            else:

                print("Invalid employee type.")

                continue

            emp\_manager.add\_employee(emp)

        elif choice == '2':

            emp\_manager.display\_employees()

        elif choice == '3':

            try:

                index = int(input("Enter the index of the employee to update: "))

                if 0 <= index < len(emp\_manager.get\_employees()):

                    emp\_type = input("Enter employee type (manager/worker): ").strip().lower()

                    name = input("Enter name: ")

                    age = int(input("Enter age: "))

                    salary = float(input("Enter salary: "))

                    if emp\_type == 'manager':

                        department = input("Enter department: ")

                        emp = Manager(name, age, salary, department)

                    elif emp\_type == 'worker':

                        hours\_worked = int(input("Enter hours worked: "))

                        emp = Worker(name, age, salary, hours\_worked)

                    else:

                        print("Invalid employee type.")

                        continue

                    emp\_manager.update\_employee(index, emp)

                else:

                    print("Invalid index.")

            except ValueError:

                print("Invalid input for index. Please enter a valid number.")

        elif choice == '4':

            try:

                index = int(input("Enter the index of the employee to delete: "))

                if 0 <= index < len(emp\_manager.get\_employees()):

                    emp\_manager.delete\_employee(index)

                else:

                    print("Invalid index.")

            except ValueError:

                print("Invalid input for index. Please enter a valid number.")

        elif choice == '5':

            break

        else:

            print("Invalid choice.")

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

    main()