Name: UMER ZIA

Roll No: SU92-BSAIM-S24-030

Task: Lab 11

Submitted To: Rasikh Ali

Lab Task 11: Employee Management System with Encapsulation, File Handling, and Menu-Driven Interface

----

```
class Employee(ABC):
    def __init__(self, name, age, salary):
        self.__name = name
        self.__age = age
        self.__salary = salary

    # Getter and Setter for name
    def get_name(self):
        return self.__name

    def set_name(self, name):
        self.__name = name

    # Getter and Setter for age
    def get_age(self):
        return self.__age

    def set_age(self, age):
        self.__age = age

    # Getter and Setter for salary
    def get_salary(self):
        return self.__salary

    def set_salary(self, salary):
        self.__salary = salary

    @abstractmethod
    def get_details(self):
        pass

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 get_details(self):
        return f"Manager: {self.get_name()}, Age: {self.get_age()}, Salary: {self.get_salary()},
Department: {self.get_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 get_details(self):
        return f"Worker: {self.get_name()}, Age: {self.get_age()}, Salary: {self.get_salary()}, Hours
Worked: {self.get_hours_worked()}"

def save_employees(employees, file_name="employee_data.csv"):
    with open(file_name, 'w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(['Name', 'Age', 'Salary', 'Department', 'Hours Worked'])
        for emp in employees:
            if isinstance(emp, Manager):
                writer.writerow([emp.get_name(), emp.get_age(), emp.get_salary(),
emp.get_department(), ''])
```

```
            elif isinstance(emp, Worker):
                writer.writerow([emp.get_name(), emp.get_age(), emp.get_salary(), '',
emp.get_hours_worked()])


def load_employees(file_name="employee_data.csv"):
    employees = []
    try:
        with open(file_name, 'r') as file:
            reader = csv.reader(file)
            next(reader)  # Skip header
            for row in reader:
                if row[3]:  # Department is not empty
                    employees.append(Manager(row[0], int(row[1]), float(row[2]), row[3]))
                elif row[4]:  # Hours worked is not empty
                    employees.append(Worker(row[0], int(row[1]), float(row[2]), int(row[4])))
    except FileNotFoundError:
        pass
    return employees

def add_employee(employees):
    employee_type = input("Enter employee type (Manager/Worker): ").strip().lower()
    name = input("Enter name: ").strip()
    age = int(input("Enter age: "))
    salary = float(input("Enter salary: "))

    if employee_type == 'manager':
        department = input("Enter department: ").strip()
        employees.append(Manager(name, age, salary, department))
    elif employee_type == 'worker':
        hours_worked = int(input("Enter hours worked: "))
        employees.append(Worker(name, age, salary, hours_worked))
    else:
        print("Invalid employee type!")
        return
    save_employees(employees)
    print("Employee added successfully!")


def display_employees(employees):
    if not employees:
        print("No employees found.")
    else:
        for emp in employees:
            print(emp.get_details())


def update_employee(employees):
    name = input("Enter the name of the employee to update: ").strip()
    for emp in employees:
        if emp.get_name() == name:
            attribute = input("Enter the attribute to update (name, age, salary, department,
hours_worked): ").strip()
            if attribute in ['name', 'age', 'salary', 'department', 'hours_worked']:
                value = input(f"Enter new value for {attribute}: ").strip()
                if attribute == 'age':
                    value = int(value)
                elif attribute == 'salary':
                    value = float(value)
                elif attribute == 'hours_worked':
                    value = int(value)
                getattr(emp, f"set_{attribute}")(value)
                save_employees(employees)
                print("Employee updated successfully!")
                return
    print("Employee not found!")


def delete_employee(employees):
    name = input("Enter the name of the employee to delete: ").strip()
    employees[:] = [emp for emp in employees if emp.get_name() != name]
    save_employees(employees)
    print("Employee deleted successfully!")


# Main menu
def main():
    employees = load_employees()

    while True:
        print("\nEmployee Management System")
        print("1. Add Employee")
        print("2. Display Employees")
        print("3. Update Employee")
        print("4. Delete Employee")
        print("5. Exit")

        choice = input("Enter your choice: ").strip()
```

```
if choice == '1':
    add_employee(employees)
elif choice == '2':
    display_employees(employees)
elif choice == '3':
    update_employee(employees)
elif choice == '4':
    delete_employee(employees)
elif choice == '5':
    break
else:
    print("Invalid choice! Please try again.")
```