Object Oriented Programming

Lab 10

Task 1; Program to manage employee personal details:

Code:

```
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 display_info(self):
        print(f"Name: {self.__name}")
        print(f"Age: {self.__age}")
        print(f"Salary: {self.__salary}")

    def get_employee_type(self):
        return "Employee"

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 display_info(self):
        super().display_info()
        print(f"Department: {self.__department}")
```

```
    def get_employee_type(self):
        return "Manager"

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 display_info(self):
        super().display_info()
        print(f"Hours Worked: {self.__hours_worked}")

    def get_employee_type(self):
        return "Worker"

def add_employee(employees):
    print("Adding a new employee:")
    name = input("Enter name: ")
    age = int(input("Enter age: "))
    salary = float(input("Enter salary: "))
    employee_type = input("Enter employee type (Manager/Worker): ").lower()
```

```
    if employee_type == "manager":
        department = input("Enter department: ")
        employee = Manager(name, age, salary, department)
    elif employee_type == "worker":
        hours_worked = float(input("Enter hours worked: "))
        employee = Worker(name, age, salary, hours_worked)
    else:
        print("Invalid employee type.")
        return

    employees.append(employee)
    print("Employee added successfully.")

def display_employees(employees):
    if not employees:
        print("No employees in the records.")
        return

    print("Employee Details:")
    for employee in employees:
        employee.display_info()
        print(f"Employee Type: {employee.get_employee_type()}")
        print("-" * 20)

def update_employee(employees):
    if not employees:
        print("No employees in the records.")
        return
```

```
    name = input("Enter name of employee to update: ")
    found = False
    for employee in employees:
        if employee.get_name() == name:
            found = True
            print("Updating employee details:")
            employee.set_name(input("Enter new name (leave blank to keep current): ") or employee.get_name())
            employee.set_age(int(input("Enter new age (leave blank to keep current): ") or employee.get_age()))
            employee.set_salary(float(input("Enter new salary (leave blank to keep current): ") or employee.get_salary()))
            if isinstance(employee, Manager):
                employee.set_department(input("Enter new department (leave blank to keep current): ") or employee.get_department())
            elif isinstance(employee, Worker):
                employee.set_hours_worked(float(input("Enter new hours worked (leave blank to keep current): ") or employee.get_hours_worked()))
            print("Employee updated successfully.")
            break

    if not found:
        print("Employee not found.")

def delete_employee(employees):
    if not employees:
        print("No employees in the records.")
        return
```

```
    name = input("Enter name of employee to delete: ")
    found = False
    for i, employee in enumerate(employees):
        if employee.get_name() == name:
            found = True
            del employees[i]
            print("Employee deleted successfully.")
            break

    if not found:
        print("Employee not found.")

def save_employees(filename, employees):
    with open(filename, 'w', newline='') as file:
        writer = csv.writer(file)
        writer.writerow(["Type", "Name", "Age", "Salary", "Department", "Hours Worked"])
        for employee in employees:
            if isinstance(employee, Manager):
                writer.writerow(["Manager", employee.get_name(), employee.get_age(),
employee.get_salary(), employee.get_department(), ""])
            elif isinstance(employee, Worker):
                writer.writerow(["Worker", employee.get_name(), employee.get_age(),
employee.get_salary(), "", employee.get_hours_worked()])

def load_employees(filename):
    employees = []
    try:
        with open(filename, 'r', newline='') as file:
```

```
            reader = csv.reader(file)
            next(reader)

            for row in reader:
                if row[0] == "Manager":
                    employees.append(Manager(row[1], int(row[2]), float(row[3]), row[4]))
                elif row[0] == "Worker":
                    employees.append(Worker(row[1], int(row[2]), float(row[3]), float(row[5])))
    except FileNotFoundError:
        print(f"File '{filename}' not found. Starting with an empty employee list.")
    return employees

def main():
    filename = 'employees.csv'
    employees = load_employees(filename)

    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: ")

        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':
            save_employees(filename, employees)
            print("Exiting program.")
            break
        else:
            print("Invalid choice.")

if __name__ == "__main__":
    main()
```

Output:

```
Employee Management System
1. Add Employee
2. Display Employees
3. Update Employee
4. Delete Employee
5. Exit
Enter your choice: 1
Adding a new employee:
Enter name: Ahmad
```

Enter age: 18

Enter salary: 100000000000

Enter employee type (Manager/Worker): Worker

Enter hours worked: 43

Employee added successfully.

Employee Management System

1. Add Employee

2. Display Employees

3. Update Employee

4. Delete Employee

5. Exit

Enter your choice: 2

Employee Details:

Name: Ayann

Age: 19

Salary: 100000.0

Department: Upper Management

Employee Type: Manager

Name: Ahmad

Age: 18

Salary: 100000000000.0

Hours Worked: 43.0

Employee Type: Worker

Employee Management System

1. Add Employee

2. Display Employees

3. Update Employee

4. Delete Employee

5. Exit

Enter your choice: 5

Exiting program.