**Name: Usman Ul haq**

**Roll number: SU92-BSAIM-S24-035**

**Section: 2A**

**Semester: 2nd**

**Task = 10**

**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).**

**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.**

FILE\_NAME = "employee\_records.csv"

import csv

class Person:

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):

return f"Name: {self.\_\_name}, Age: {self.\_\_age}, Salary: {self.\_\_salary}"

class Supervisor(Person):

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):

return super().display\_info() + f", Department: {self.\_\_department}"

class Laborer(Person):

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):

return super().display\_info() + f", Hours Worked: {self.\_\_hours\_worked}"

FILE\_NAME = "employees.csv"

def initialize\_csv():

try:

with open(FILE\_NAME, "x", newline="") as file:

writer = csv.writer(file)

writer.writerow(["Type", "Name", "Age", "Salary", "Department", "Hours Worked"])

except FileExistsError:

pass

def add\_employee(employee):

with open(FILE\_NAME, "a", newline="") as file:

writer = csv.writer(file)

if isinstance(employee, Supervisor):

writer.writerow(["Supervisor", employee.get\_name(), employee.get\_age(), employee.get\_salary(), employee.get\_department(), ""])

elif isinstance(employee, Laborer):

writer.writerow(["Laborer", employee.get\_name(), employee.get\_age(), employee.get\_salary(), "", employee.get\_hours\_worked()])

def display\_all\_employees():

try:

with open(FILE\_NAME, "r") as file:

reader = csv.reader(file)

for row in reader:

print(", ".join(row))

except FileNotFoundError:

print("No employee records found.")

def update\_employee(name, field, value):

updated = False

try:

with open(FILE\_NAME, "r") as file:

rows = list(csv.reader(file))

for row in rows:

if row[1] == name:

if field == "age":

row[2] = value

elif field == "salary":

row[3] = value

elif field == "department" and row[0] == "Supervisor":

row[4] = value

elif field == "hours\_worked" and row[0] == "Laborer":

row[5] = value

updated = True

with open(FILE\_NAME, "w", newline="") as file:

writer = csv.writer(file)

writer.writerows(rows)

if updated:

print(f"Employee {name}'s {field} updated successfully.")

else:

print(f"Employee {name} not found or invalid field for employee type.")

except FileNotFoundError:

print("No employee records found.")

def delete\_employee(name):

try:

with open(FILE\_NAME, "r") as file:

rows = list(csv.reader(file))

updated\_rows = [row for row in rows if row[1] != name]

with open(FILE\_NAME, "w", newline="") as file:

writer = csv.writer(file)

writer.writerows(updated\_rows)

if len(rows) > len(updated\_rows):

print(f"Employee {name} deleted successfully.")

else:

print(f"Employee {name} not found.")

except FileNotFoundError:

print("No employee records found.")

initialize\_csv()

while True:

print("1. Add New Employee")

print("2. Display All Employees")

print("3. Update Employee Information")

print("4. Delete Employee")

print("5. Exit")

choice = input("Enter your choice (1-5): ").strip()

if choice == "1":

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

name = input("Enter name: ").strip()

age = input("Enter age: ").strip()

salary = input("Enter salary: ").strip()

if emp\_type == "supervisor":

department = input("Enter department: ").strip()

supervisor = Supervisor(name, age, salary, department)

add\_employee(supervisor)

elif emp\_type == "laborer":

hours\_worked = input("Enter hours worked: ").strip()

laborer = Laborer(name, age, salary, hours\_worked)

add\_employee(laborer)

else:

print("Invalid employee type. Please enter 'Supervisor' or 'Laborer'.")

elif choice == "2":

display\_all\_employees()

elif choice == "3":

name = input("Enter the name of the employee to update: ").strip()

field = input("Enter the field to update (age/salary/department/hours\_worked): ").strip()

value = input("Enter the new value: ").strip()

update\_employee(name, field, value)

elif choice == "4":

name = input("Enter the name of the employee to delete: ").strip()

delete\_employee(name)

elif choice == "5":

break

else:

print("Invalid choice. Please enter a number between 1 and 5.")