**Name: Usman Ul haq**

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

**Section: 2A**

**Semester: 2nd**

**Task = 11**

**Program to manage employee personal details:**

**Same as Task 10. Just implementing the DATA ABSTRACTION in the task**

FILE\_NAME = "employee\_records.csv"

import csv

from abc import ABC, abstractmethod

FILE\_NAME = "employee\_records.csv"

class Employee(ABC):

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

@abstractmethod

def display\_info(self):

pass

class Supervisor(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):

return f"Name: {self.get\_name()}, Age: {self.get\_age()}, Salary: {self.get\_salary()}, Department: {self.\_\_department}"

class Laborer(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):

return f"Name: {self.get\_name()}, Age: {self.get\_age()}, Salary: {self.get\_salary()}, Hours Worked: {self.\_\_hours\_worked}"

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)

return list(reader)

except FileNotFoundError:

return []

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)

return updated

except FileNotFoundError:

return False

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)

return len(rows) > len(updated\_rows)

except FileNotFoundError:

return False

**Benefits of Data Abstraction:**

* **Clear Interface**: The user of the Employee, Supervisor, and Laborer classes only needs to know about the interface (i.e., methods such as display\_info) and not the internal implementation details.
* **Separation of Concerns**: Each subclass (Supervisor and Laborer) is only responsible for its own specific behavior (e.g., display\_info), making the system easier to maintain and extend.