**EmployeeManagementSystem.**

**Contents:**

* **main.py**
* **employee.py**
* **payroll\_system.py**
* **README.md**
* **LICENSE**

**Code**

**main.py**

import payroll\_system

def main():

payroll\_system = payroll\_system.PayRollSystem()

while True:

payroll\_system.display\_menu()

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

payroll\_system.process\_user\_choice(choice)

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

main()

**employee.py**

from abc import ABC, abstractmethod

class Employee(ABC):

"""Represents an employee."""

def \_\_init\_\_(self, name: str, id: int):

"""Initializes an employee with a name and ID."""

self.name = name

self.id = id

@abstractmethod

def calculate\_salary(self) -> float:

"""Calculates the employee's salary."""

pass

def \_\_str\_\_(self) -> str:

"""Returns a string representation of the employee."""

return f"Employee[name={self.name}, id={self.id}, Salary={self.calculate\_salary():.2f}]"

class FullTimeEmployee(Employee):

"""Represents a full-time employee."""

def \_\_init\_\_(self, name: str, id: int, monthly\_salary: float):

"""Initializes a full-time employee with a name, ID, and monthly salary."""

super().\_\_init\_\_(name, id)

self.monthly\_salary = monthly\_salary

def calculate\_salary(self) -> float:

"""Calculates the full-time employee's salary."""

return self.monthly\_salary

def set\_monthly\_salary(self, monthly\_salary: float) -> None:

"""Sets the full-time employee's monthly salary."""

self.monthly\_salary = monthly\_salary

class PartTimeEmployee(Employee):

"""Represents a part-time employee."""

def \_\_init\_\_(self, name: str, id: int, hours\_worked: float, hourly\_rate: float):

"""Initializes a part-time employee with a name, ID, hours worked, and hourly rate."""

super().\_\_init\_\_(name, id)

self.hours\_worked = hours\_worked

self.hourly\_rate = hourly\_rate

def calculate\_salary(self) -> float:

"""Calculates the part-time employee's salary."""

return self.hours\_worked \* self.hourly\_rate

def set\_hourly\_rate(self, hourly\_rate: float) -> None:

"""Sets the part-time employee's hourly rate."""

self.hourly\_rate = hourly\_rate

def get\_hours\_worked(self) -> float:

"""Returns the part-time employee's hours worked."""

return self.hours\_worked

**payroll\_system.py**

class PayRollSystem:

def \_\_init\_\_(self):

self.employee\_list = []

def add\_employee(self, employee):

if not self.get\_employee\_by\_id(employee.id):

self.employee\_list.append(employee)

else:

print("\nID Already exists, Cannot Add Employee With Same Duplicate ID.")

def remove\_employee(self, id):

employee\_to\_remove = self.get\_employee\_by\_id(id)

if employee\_to\_remove:

self.employee\_list.remove(employee\_to\_remove)

print("\033[92m" + "Employee removed successfully." + "\033[0m")

else:

print("\n\033[91m" + "Employee with ID " + str(id) + " not found. Cannot remove." + "\033[0m")

def adjust\_salary(self):

id = int(input("Enter employee ID: "))

employee = self.get\_employee\_by\_id(id)

if employee:

percentage = float(input("Enter percentage adjustment: "))

current\_salary = employee.calculate\_salary()

adjustment\_amount = current\_salary \* (percentage / 100)

new\_salary = current\_salary + adjustment\_amount

if isinstance(employee, FullTimeEmployee):

employee.set\_monthly\_salary(new\_salary)

elif isinstance(employee, PartTimeEmployee):

employee.set\_hourly\_rate(new\_salary / employee.get\_hours\_worked())

print("\033[92m" + "Salary adjusted successfully for employee with ID " + "\033[0m" + str(id))

else:

print("\033[91m" + "Employee with entered ID not found. Cannot adjust salary." + "\033[0m")

def display\_employees(self):

for employee in self.employee\_list:

print("\n" + str(employee))

def get\_employee\_by\_id(self, id):

for employee in self.employee\_list:

if employee.id == id:

return employee

return None

def display\_menu(self):

print("\n \033[93m" + "\t\tMenu: \n")

print(" Employee Management System: \n")

print("1. Remove Employee")

print("2. Adjust Salary")

print("3. Display Employees")

print("4. Add employee")

print("5. Search employee")

print("0. Exit \n")

def process\_user\_choice(self, choice):

if choice == 1:

id\_to\_remove = int(input("Enter employee ID to remove: "))

self.remove\_employee(id\_to\_remove)

elif choice == 2:

self.adjust\_salary()

elif choice == 3:

print("\nEmployee Details: ")

self.display\_employees()

elif choice == 4:

self.add\_employee(self.create\_employee())

elif choice == 5:

self.search\_employee\_by\_id()

elif choice == 0:

print("Exiting program. Goodbye!")

else:

print("\033[91m" + "Invalid choice. Please enter a valid option." + "\033[0m")

def create\_employee(self):

name = input("Enter employee name: