22p-9318_Ahmed_BS(SE)-5B

September 7, 2024

[1]:
```
class Employee:
    def __init__(self, name, age, salary, position, hire_date):
        #constructor
        self.name=name
        self.age=age
        self.salary=salary
        self.position=position
        self.hire_date=hire_date

    def __del__(self):
        #destructor
        print(f"{self.name} has been removed from the system")

    def __str__(self):
        #string representation
        return f"Employee: {self.name}, Position: {self.position}"

    def __repr__(self):
        #representation of employee object
        return f"Employee({self.name}, {self.age}, {self.salary}, {self.
 ↪position})"

    def display_details(self):
        #displaying details
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")
        print(f"Salary: {self.salary}")
        print(f"Position: {self.position}")
        print(f"Hire Date: {self.hire_date}")

    def annual_bonus(self):
        #I assigned annual bonus as 10% of the salary
        return self.salary*0.10

    def promote(self, new_position, increment):
        #promoting and increasing salary of employee
        self.position=new_position
        self.salary=self.salary+increment
        print(f"{self.name} has been promoted to {self.position} with a salary␣
↪increment of {increment}")

    def demote(self, new_position, decrement):
        #demoting salary and position, here threshold is added by me for having␣
↪a meaningful logic
        new_salary=self.salary-decrement
        if new_salary>=30000:
            self.position = new_position
            self.salary=new_salary
            print(f"{self.name} has been demoted to {self.position} with a new␣
↪salary of {self.salary}")
        else:
            print(f"Cannot demote {self.name} as it would drop their salary␣
↪below the allowed threshold")

    def retirement_age(self):
        #years left until retirement
        return 60-self.age

    def increase_salary(self, percentage):
        #increasing salary
        increment=(lambda sal:sal+(sal*percentage/100))(self.salary)
        self.salary=increment
        print(f"{self.name}'s salary increased by {percentage}% to {self.
↪salary}")

    def compare_salary(self, other_employee):
        #comparing salary with other employee
        if self.salary>other_employee.salary:
            return "Higher"
        elif self.salary<other_employee.salary:
            return "Lower"
        else:
            return "Equal"
```

[2]:
```
class Manager(Employee):
    def __init__(self, name, age, salary, position, hire_date, team_size):
        #constructor
        super().__init__(name, age, salary, position, hire_date)
        self.team_size=team_size

    def __str__(self):
        #string representation of the manager class, including the team size
        return f"Manager: {self.name}, Team Size: {self.team_size}"
    def annual_bonus(self):
        #overriding bonus calculation: 15% of salary + 1% per team member
        return self.salary*0.15+self.salary*0.01*self.team_size

    def increase_team_size(self, new_members):
        #increasing team size by adding new members
        self.team_size=self.team_size+new_members
        print(f"{self.name}'s team size increased by {new_members} to {self.
↪team_size}")

    def reduce_team_size(self, lost_members):
        #reducing team size
        self.team_size=self.team_size-lost_members
        print(f"{self.name}'s team size reduced by {lost_members} to {self.
↪team_size}")
```

[3]:
```
def update_employee_info(employee, *args, **kwargs):
    if args:
        #*args to update position and salary
        employee.position=args[0]
        employee.salary=args[1]

    if kwargs:
        #**kwargs to update other attributes like name, age, hire_date
        for key, value in kwargs.items():
            setattr(employee, key, value)

#total annual cost (salary + bonus) for all employees
def total_annual_cost(*employees):
    total_cost=0
    for employee in employees:
        total_cost=total_cost+employee.salary*12+employee.annual_bonus()
    return total_cost
```

[4]:
```
first_employee=Employee("Ahmed", 20, 200000, "Cyber Expert", "18-01-2019")
second_employee=Employee("Ali", 22, 300000, "Business Analyst", "21-03-2016")
senior_manager=Manager("Anas", 26, 600000, "Team Lead", "10-07-2012", 9)

# Display initial details
print("\nInitial employee details:")
first_employee.display_details()
second_employee.display_details()
senior_manager.display_details()

# Promotions and Demotions
first_employee.promote("Cyber Pro", 100000)  # Promotion increment of 100000
```

```
second_employee.demote("Junior Business Analyst", 100000)  # Demotion decrement␣
↪of 100000

# Salary increase for second employee
second_employee.increase_salary(5)  # Increase salary by 5%

# Manager-specific actions
senior_manager.increase_team_size(3)  # Increase team size by 3 members

# Dynamic update of employee information
update_employee_info(first_employee, "Lead Cyber Expert", 350000, name="Ahmed␣
↪Ali", age=21)

# Display updated details
print("\nUpdated employee details:")
first_employee.display_details()
second_employee.display_details()
senior_manager.display_details()

# Calculate and display total annual cost for all employees
total_cost=total_annual_cost(first_employee, second_employee, senior_manager)
print(f"\nTotal annual cost: {total_cost}")
```

```
Initial employee details:
Name: Ahmed
Age: 20
Salary: 200000
Position: Cyber Expert
Hire Date: 18-01-2019
Name: Ali
Age: 22
Salary: 300000
Position: Business Analyst
Hire Date: 21-03-2016
Name: Anas
Age: 26
Salary: 600000
Position: Team Lead
Hire Date: 10-07-2012
Ahmed has been promoted to Cyber Pro with a salary increment of 100000
Ali has been demoted to Junior Business Analyst with a new salary of 200000
Ali's salary increased by 5% to 210000.0
Anas's team size increased by 3 to 12

Updated employee details:
Name: Ahmed Ali
```

```
Age: 21
Salary: 350000
Position: Lead Cyber Expert
Hire Date: 18-01-2019
Name: Ali
Age: 22
Salary: 210000.0
Position: Junior Business Analyst
Hire Date: 21-03-2016
Name: Anas
Age: 26
Salary: 600000
Position: Team Lead
Hire Date: 10-07-2012

Total annual cost: 14138000.0
```

```
[ ]:
```