Inspiring Excellence

Course Code:	**CSE111**
Course Title:	**Programming Language II**
Homework No:	**06**
Topic:	**Inheritance**
Submission Type:	**Submission Link:** https://docs.google.com/forms/d/e/1FAIpQLSdnzY40cOuNvzni4q2d2GlQhLvaZuYtZldn26Yue2_K9WOd5w/viewform
Resources:	1. **Class lectures** 2. **BuX lectures** a. **English:** i. **Inheritance: here**

	b. Supplementary: **i. Inheritance: here**

Task 1

Given below are four classes, which follow the Inheritance concept of OOP. The parent/base class here is the Employee class, from which two child classes have been derived, namely Programmer and HR. Again from the Programmer class, another class named InternProgrammer has been derived.

Employee Class

Class Variable	Description
employee_count : dictionary	It will keep track of all types of employees (except intern programmers) being created. This dictionary will initially remain empty. For every Programmer and HR object being created, this dictionary will be filled with 'Programmer' and 'HR' as the keys and their corresponding object counts as values. For example, the dictionary will be empty {} at first. Now if a Programmer and HR object is created, the dictionary will look like this: {'Programmer' : 1, 'HR' : 1}

Instance Variable	Description
name : string	Stores the name of the employee
joining_date : string	Stores the date on which the employee joined the company, in the format "YYYY-MM-DD" e.g. "2020-06-21"
work_experience : integer	Stores the previous work experience of an employee in years
weekly_work_hour : integer	Stores the weekly work hour of an employee. The default value for this variable should be 40 hours. But the weekly work hour should not exceed 60 hours. In case of an invalid value, it should display an error message and set the variable to 40.

Class Method	Description
showDetails()	A method that prints both the total employee count and the individual employee count (except intern programmers). For example, Total Employee/s : 55 Total Programmer/s: 40 Total HR Employee/s: 15

Programmer Class

Class Variable	Description
designation_list : list	A list containing all the designations or roles that a programmer can be assigned. Elements of this list are ['Junior Software Engineer', 'Software Engineer', 'Senior Software Engineer', 'Technical Lead']. The lowest to highest index of this list indicates the gradual increase in the designation of the Programmers.

Instance Variable	Description
	inherit **name**, **joining_date**, **work_experience** and **weekly_work_hour** from Employee class
id : string	Stores an employee ID, which must be created from the constructor using the createProgrammerID() method
designation : string	Determines the designation of a programmer from the work_experience variable. Work experience: Designation 0 to < 3 years: Junior Software Engineer 3 to < 5 years: Software Engineer 5 to < 8 years: Senior Software Engineer > 8 years: Technical Lead

<table>
<tr><th>Instance Method</th><th>Description</th></tr>
<tr><td>showProgrammerDetails()</td><td>A method that prints the programmer name, ID, joining_date (in DD-MM-YYYY format), designation and the salary that s/he receives</td></tr>
<tr><td>calculateSalary()</td><td>A method to calculate the salary of a programmer, based on their designation and the number of years they have been employed in the company.
<table>
<tr><th>Designation</th><th>Base Salary</th></tr>
<tr><td>Junior Software Engineer</td><td>BDT 30,000</td></tr>
<tr><td>Software Engineer</td><td>BDT 45,000</td></tr>
<tr><td>Senior Software Engineer</td><td>BDT 70,000</td></tr>
<tr><td>Technical Lead</td><td>BDT 120,000</td></tr>
</table>

The salary will be incremented by 15% every year. For example, if the base salary is 45,000, then by the end of 1 year, the salary will be 51,750. Again, by the end of 2 years, the salary will be 15% of 51,750, which is 59,512.5 and so on</td></tr>
<tr><td>createProgrammerID()</td><td>A method that generates an ID for a programmer using their joining_date, total employee count (Employee class) and the type of employee (Programmer in this case) and returns the generated string. For example, if a Programmer joined the company on ‘2020-04-05’ and the total number of employees is 25, the generated ID should be ‘P-200405-25’</td></tr>
<tr><td>calculateOvertime()</td><td>A method to calculate how many hours a programmer has worked over the weekly work hour limit. For every extra hour worked, the programmer will receive BDT 500. Assume that every month has 4 weeks. The amount received from overtime will be added to the salary, so make sure to calculate accordingly. This calculated amount should be printed as well. For example, 10 hours worked above the weekly work hour limit would generate BDT 20,000, which is to be added to the monthly salary of the Programmer</td></tr>
</table>

HR Class

Instance Variable	Description
	inherit **name**, **joining_date**, **work_experience** and **weekly_work_hour** from Employee class
id : string	Stores an HR employee ID, which must be created from the constructor using the createHREmployeeID() method

Instance Method	Description
showHREmployeeDetails()	A method that prints the HR employee name, ID and joining_date
createHREmployeeID()	A method that generates an ID for an HR employee using their joining_date, total employee count (Employee class) and the type of employee (HR in this case) and returns the generated string. For example, if an HR employee joined the company on '2020-04-05' and the total number of employees is 25, the generated ID should be 'HR-200405-25'

InternProgrammer Class

Class Variable	Description
intern_count : integer	It will keep track of the number of intern_programmer objects being created

Instance Variable	Description
temp_id : integer	A temporary ID that is to be generated using the class variable. Since the intern is not a full-time employee, they will not be given IDs following the previous format. For example, if the current number of intern programmers in the company is 5, then the intern will be assigned a temporary ID following this format: 'Temp_5'
intern_type : string	A variable to define whether an intern is paid or unpaid. The default value is 'Unpaid'
work_experience : integer	The work experience of an intern will always be 0
weekly_work_hour : integer	The weekly work hour for an intern will always be fixed at 40 hours
joining_date : string	Inherit this variable from its parent class
status : string	A flag to check whether an intern is eligible for promotion to a Programmer or not. If the intern has completed 4 months in the company, then s/he will be eligible for promotion. Keep in mind that interns are only recruited in January and July of a particular year.

Instance Method	Description
showInternDetails()	A method that prints the intern programmer name, temporary ID, joining_date, the type of intern they are and their eligibility status
promoteToProgrammer()	Using the status variable, this method will help determine if an intern is eligible for promotion to a Programmer or not. If they are eligible, then an object of Programmer class is to be created and returned. Before returning, it should also display a message saying, "The intern is promoted!". Otherwise, a

	message will be displayed saying “The intern can not be promoted”. For example, if the intern has been working in the company for less than 4 months, then they are still an intern; otherwise they will be promoted to a Programmer. The newly promoted programmer can be considered to have no experience, 40 weekly work hours and joining date should be the day promoteToProgrammer() method is called. For example, if promoteToProgrammer() method is called on 16th August, 2023, joining date should be “2023-08-16”

Task:

Your task is to design all the four classes using Inheritance as well as the driver code. You cannot use concepts of overriding to complete this task. Make sure that all the methods mentioned above are working as described.

A demo driver along with output can be found in this colab file