Object-Oriented Programming Lab#10, Fall 2022

Today's Topics

- Inheritance
- polymorphism
- abstract class
- exception

An Employee Record System

Create an Employee Record System for "UAP HR" department. The application will be used by both the **admin** and **the employee**. Depending on the role of the user (admin vs. employee), different functionalities will be exposed. Admin will be able to add/edit/view any information of any employee whereas the employee will have mainly the view functionalities and edit capability for his own information. Each Employee is identified by **his/her name, employee id and position/designation**. There could be 3 types of employees;

1) **Salaried employee** -> this type of employees is paid a fixed monthly salary regardless of the number of hours worked.
2) **Hourly employee** -> They are paid by the hour. They have an hourly rate and their payment will depend on how many hours they worked. The more they work, the more they will be paid. So, the salary will be [hour worked per month* hourly rate].
3) **Commission employee**-> They are paid a percentage of their sales. If their percentage (known as **commission**) is "a" and total monthly sale is" b", the total monthly salary will be [a*b/100];

Note**: For all 3 employees, **increase salary feature/functionality will increase the rate not the total salary**. Set the rate of each employee will do the following

a. for **salaried** employee it would set the **monthly** rate,
b. for **hourly** employee it will set the **hourly rate** and
c. for **commission** employee it will set the **percentage**.

Here is the list of functionalities of the system.

2) A user can log in as an admin or employee. For simplicity, we can **skip** the log-in part and add an option or button for logging in as an **admin** or **employee**.
3) Admin will have the following functionalities
    a. add new Employee to the system,
    b. increase the salary/rate of an employee
    c. view the monthly salary of any employee,
    d. view the details of a specific employee
    e. view the details of all employees
    f. view the monthly record of an hourly/commission employee
    g. view the monthly record of a commission employee

4) Employee will have the following functionalities
    a. view his details
    b. edit phone no
    c. search for an employee by id
    d. search for an employee by name
    e. enter today's record

[Note: As part of today's lab, you need to complete all red marked section.]

Object-Oriented Programming Lab#10, Fall 2022

Implementation

Note: The following design is just one possible option. You can modify it according to your need. You are free to add additional attributes, methods.

You need 2 projects for this Lab.

Create a Project name it as you want (and do the following) : ***(Note: Do not use default package)***

1) Create the following Exception class. Note: Use the same package as the rest of the class.

```
public class InvalidEmployeeException extends Exception {

 public InvalidEmployeeException(String id) {
  super(String.format("Employee with Id:%s is not a valid employee.", id));

 }

}
```

2) Create a class name **DailyRecord**.
   a. Add attributes: **LocalDate** date and double hour_Or_Sale;
   b. Add Constructor public DailyRecord(double hour_Or_Sale)
   - Assign date to LocalDate.now() and initialize hour_or_sale with the parameter.
   c. Add Constructor public DailyRecord(String date, double hour_Or_Sale)

   Code to convert String to LocalDate

   ```
   DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");
   this.date = LocalDate.parse(date,formatter);
   ```

   d. Override public String toString()
   - Return the time and hour/sale

   Code to convert LocalDate to String

   ```
   DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy");
   String appTime = date.format(formatter);
   ```

3) Create an **abstract** class **Employee** which has **5 private instance variables*; name, id, age, phoneNum, designation***.

   a. Create a **constructor** that takes initial values for all attributes except **id** and initializes those attributes. Also, generate a 4-digit random number and assign that to the id instance variable.

   Create the **following methods as** described

   a. Create public getter/setter method for all these attributes.
   b. *Override **public String toString()** method*

- This method should return a String in the format "Name]: ***name***; Id:***id***; Email:email; Contact no:***phoneNo***; Designation:***designation***;" where ***name, id, age, phoneNum,*** and ***designation*** are the attributes of the Employee class. You can return in a different format if you want.

c. Define the following abstract methods.

- ***public abstract double getSalary()***
- ***public abstract void increaseSalary(double amt)***
- ***public abstract String toString(boolean details)***

4) Create a concrete **SalariedEmployee** class and make it the subclass of **Employee** class. Now do the following.

a. Add a new attribute name ***monthlySalary***.

d. Implement parameterized constructor. You need to pass all attributes including the monthly salary but not the id. Call the parent's constructor and initialize the ***monthlySalary***. Also add "11-" as the prefix of the id that is generated from parent class.

Override the **following 3 methods as** described

b. ***public void increaseSalary(double amt)***

- Inside the method, increase the ***monthlySalary*** by ***amt*** amount.

c. ***public double getSalary()*** method

- Returns the ***monthlySalary*** *value*.

e. ***public String toString(boolean details)*** *method*

- if details is false call & return the parent's toString() method. Otherwise, call the ***toString()*** method of parent class, concatenate "; Salary: ***monthlySalary***" and return the concatenated string.

5) Create another **abstract** class **PartTimeEmployee** and make it a subclass of **Employee** class.

a. Add an attribute ArrayList<**DailyRecord**> **dailyRecords** and use the protected access modifier.

b. Create constructor.

c. Add following methods.

i. public **abstract** void addRecord(DailyRecord record)

ii. public **abstract** void addRecord(double hour_Or_Sale)

iii. public ArrayList<DailyRecord> getDailyRecords()

6) Create **HourlyEmployee** class and make it the subclass of **PartTimeEmployee** class. Now do the following.

a. Add 2 new attributes name ***hourlyRate*** *and* ***hourWorkedPerMonth***.

b. Create a **constructor** that takes initial value for all attributes except the ***hourWorkedPerMonth*** and initializes those attributes. Also add "22-" as the prefix of the id that is generated from parent class.

- Inside the constructor, call parent class's constructor. Also set the ***hourlyRate***.

c. Create public getter/setter method for ***hourWorkedPerMonth***.

d. Create a method ***public double getSalary(int hWorked)***

- Returns the total payment/salary which will be [h_worked* hourly rate**]**.

Override the **following methods as** described

e. ***public void increaseSalary(double amt)***

- Inside the method, increase the ***hourlyRate*** by ***amt*** amount.

f. ***public double getSalary()***

- Returns the total payment/salary which will be [hour worked per month* hourly rate**]**.

g. ***public void addRecord(DailyRecord record)***

- Add the **record** to **dailyRecords** attribute and increase the **hourWorkedPerMonth** by the **hour_or_sale** attribute of record parameter.

h. ***public void addRecord(double hour_or_sale)***

- Create a **DailyRecord** object using the **hour_or_sale** parameter and add the object to **dailyRecords** attribute. Also, increase the **hourWorkedPerMonth** by the **hour_or_sale** attribute of record parameter.

i. ***public String toString(boolean details)*** *method*

- if details is false call & return the parent's toString() method. Otherwise, call the ***toString()*** method of parent class, concatenate "; Rate:***hourlyRate***" and return the concatenated string.

7) Create **CommissionEmployee** class and make it the subclass of **PartTimeEmployee** class. Now do the following.

a. Add 2 new attributes name ***commission*** *and* ***sale***.

b. Create a **constructor** that takes initial value for those 4 attributes except ***sale*** and initializes those attributes.

- Inside the constructor, call parent class's constructor. Also set the ***commission***. Also add "33-" as the prefix of the id that is generated from parent class.

c. Create public getter/setter method for ***sale***.

d. Create a method ***public double getSalary(double _sale)***

- Returns the total payment/salary which will be ***commission*** * ***_sale***.

Override the **following 3 methods as** described

e. ***public void increaseSalary(double amt)***

- Inside the method, increase the ***commission*** by ***amt*** amount.

f. ***public double getSalary()***

- Returns the total payment/salary which will be [***commission*** * ***sale***].

g. ***public void addRecord(DailyRecord record)***

- Add the **record** to **dailyRecords** attribute and increase the **sale** by the **hour_or_sale** attribute of record parameter.

h. ***public void addRecord(double hour_or_sale)***

- Create a **DailyRecord** object using the **hour_or_sale** parameter and add the object to **dailyRecords** attribute. Also, increase the **sale** by the **hour_or_sale** attribute of record parameter.

i. ***public String toString(boolean details)*** *method*

- if details is false call & return the parent's toString() method. Otherwise, call the ***toString()*** method of parent class, concatenate "; Commission:***commission***" and return the concatenated string.

8) Now create another class **Company** to represent the CSE department which has a list of Employee. So, there will be one attribute of type Employee ArrayList to represent the list of the employee [**ArrayList<Employee> employees**] and another attribute [name it as **name**] to store the name of the department. Add the following methods to this class.

a. **private void addNewEmployee(Employee e)**

- Add the **Employee** e to ***employees*** array.

b. ***public*** **String addSalariedEmployee(String name, ~~String employeeId,~~ int age, String designation, String phoneNo,double monthlySalary)**

- Create a **SalariedEmployee** object using the parameter provided and add the object to ***employees*** array by calling the **addNewEmployee(Employee e)** method. Also, return the id of the employee.

c. ***public*** **String addHourlyEmployee(String name~~, String employeeId~~, int age, String designation, String phoneNo,double hourlyRate)**

d. Create an **HourlyEmployee** object using the parameter provided and add the object to ***employees*** array by calling the **addNewEmployee(Employee e)** method. Also, return the id of the employee.

e. ***public*** **String addCommissionEmployee(String name, ~~String employeeId,~~ int age, String designation, String phoneNo,double commission)**

- Create an **CommissionEmployee** object using the parameter provided and add the object to ***employees*** array by calling the **addNewEmployee(Employee e)** method. Also, return the id of the employee.

f. ***public*** **Employee findEmployee(String id) throws InvalidEmployeeException**

- Loop though the ***employees*** and find the **Employee** whose id matches with the parameter provided. If the employee is found, return the employee. Otherwise, throw InvalidEmployeeException and pass the id as the parameter.

g. ***public*** **void increaseSalary(String id, double amt) throws InvalidEmployeeException**

- Find the Employee using the ***findEmployee*** method and call the ***increaseSalary(…)*** method for that object.

h. ***public*** **double getSalary(String id) throws InvalidEmployeeException**

- Find the Employee using the ***findEmployee*** method and call the ***getSalary()*** method for that object

i. ***public ArrayList<Employee> getEmployees*()**

- gettter method of ***employees***.

j. ***public ArrayList<Employee> getEmployees(String name)***

- Find all the employees whose name **partially** match with the name parameter, add all those employees in an ArrayList and return the ArrayList.

<u>Now create a new project EmployeeRecordApp (and do the following)</u>.

1. Add project reference of the previous project.
2. Create an **application class** (that has the main method) named "**EmployeeApp**" which will have the **main** method.
    - In the main method, create an object of **Company** class assign the reference to ***uap***. Ask if the user is an admin or an employee.
    - If the user is an admin show the following menu. Depending on the menu take necessary user input and call appropriate method using the ***uap*** variable.
        - add new Employee to the system,
            - Can add Salaried, Hourly, or Commission employee.
        - increase the salary/rate of an employee
        - view the monthly salary of any employee,
        - view the details of a specific employee
        - view the details of all employees
        - view the monthly record of an hourly/commission employee
        - switch role (until we save data in file)
        - exit
    - If the user is an employee, ask for his/her id. If the id is valid, show the following employee Menu. Depending on the menu take necessary user input and call appropriate method using the ***uap*** variable.
        - view profile
        - edit phone no
        - search for an employee by id
        - search for an employee by name
        - enter daily record (only for hourly and commission employee)
        - switch role (until we save data in file)
        - exit

    Note:
    1) As admin you can view everything. Hence, use toString(true) for admin. Employee can only view the details of his profile for others only summary [toString(false)] should be used.
    2) You can add an additional menu to switch user role. Otherwise, it will not be possible to show Employee menu as we are not saving data.