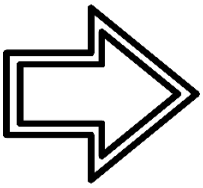
Assignment

Overview

The second iteration of our Employee Manager program(last week) is looking better. But The "We Really Suck" Vacuum Company needs a little more functionality. The company has 2 goals to keep the business running. That is selling vacuums and also repairing them. So with that in mind they need to keep track of what type of employees they have. They have 3 types to be exact:

- Sales - works off a commission rate of their weekly sales
- Repair - hourly based with a set pay rate
- Management - Salary based which have a year pay that is just split for each week.

In order to make our employee system work better we need to include these additions to our system. Since we already have the bones of an employee built we can use it as a base abstract class to which we can extend from in our other class.



Directions

To Get Started:

1. Then create a project in your IDE(IntelliJ, Eclipse, etc) titled YOURNAME-Assignment 9
2. Download and import into your project this file: **A9EmployeeChecker.java** This file will be the runner to check your work. Reference the running output at the end of the assignment.
3. Open up the employee checker file and review the code. This is the file that creates some base employees and runs through the system using the hourly, salary, and commission employee classes that YOU will create for this assignment.
4. Lastly you need the **Employee.java** & **EmergencyContact.java** file from last week. Make a copy of it and add it into your project's source file.
5. Optional: here is the updated **EmployeeSystemV3.java** file that you can run and "play" around with.

Part 1: Modify Employee.java

Reference the new UML Diagram for Employee

NOTE: The # symbol means the variable is "protected" as the access modifier. Also if it is underlined, that means it is static.

abstract Employee.java
-firstName : String -lastName : String -employeeNum : int -department : String -jobTitle : String -ArrayList<EmergencyContact> emergencyContact #currency: NumberFormat #percent : NumberFormat
<<constructor>>Employee(String fn, String ln, int en, String dept, String job) <<constructor>>Employee(String fn, String ln, int en) <<constructor>>Employee(Employee e) <<constructor>>Employee() +getFirstName() : String +setFirstName(String fn) : void +getLastName() : String +setLastName(String ln) : void +getEmployeeNumber() : int +setEmployeeNumber(int en) : void +getDepartment() : String +setDepartment(String dept) : void +getJobTitle() : String +setJobTitle(String job) : void +printEmployee() : void +printEmergencyContacts() : void +clearContacts : void +addNewContact(EmergencyContact contact) : void +getEmergencyContactList() : ArrayList<EmergencyContact> +removeContact(EmergencyContact contact) : boolean +removeContact(int index) : boolean +**abstract** resetWeek() : void +**abstract** calculateWeeklyPay() : double +**abstract** annualRaise() : void +**abstract** holidayBonus() : double +**abstract** setPay(double Pay) : void +toString() : String +equals(Object obj2) : boolean

Note the following changes:

The class is now an abstract.

We have eliminated the following fields and their methods to get/set.

- hoursWorked
- payRate

Consequently we can't call the calculateWeeklyPay() or resetHours() method anymore as we don't have the variables, BUT we still need these methods in the sub classes. So to make sure they are there, we have them as abstract methods. NOTE: Since we are going to salary and commission employees those aren't "hourly" so resetHours() doesn't make sense for a method name, instead that is switched to "resetWeek()" Again though, you are not coding it here, just defining it as abstract. We also added annualRaise() and holidayBonus() as two other abstract methods.

Also note the first constructor, we don't have a payRate to send so we get rid of the "double pr" in the parameter list. Secondly, our other constructors hoursWorked and payRate to 0 each. You need to delete those in each of the constructors. Basically once you delete those fields in your class, you should see the errors for what you need to fix/delete.

Add in the following methods to @Override

- equals
- toString()

We have added 2 static protected variables that are NumberFormat objects. These will each be set to a default currency and percent NumberFormat object.

- currency
- percent

We now have can have our NumberFormat variables protected and static to help us in our sub classes. Make that change to the currency and ADD a percent formatter. These can then be used on the Hourly, Salary, and Commission employees as needed.

  - protected static NumberFormat currency = NumberFormat.getCurrencyInstance();
  - protected static NumberFormat percent = NumberFormat.getPercentInstance();

We will have a problem with the percentage NumberFormat because it will round our percentages to a whole percent and drop the decimals. This is something we don't want

to happen on our things like the rate for Commission Employee below. So one last thing to do is to setup the percent variable to have 3 decimal places. To do this you will need to **add this in each constructor** in the Employee class:

```
percent.setMaximumFractionDigits(3);
```

New Methods:

equals(Object obj) - This method Overrides the Object equals(). It will check that the object is an Employee type, convert it and then compare the employeeNum field to the current employeeNum. If they are the same, it will return true otherwise false.

toString() - This method will return the same String contained in the printEmployee() method. You will want to refactor your code by making your print's saved into a single String with '\n' characters as needed. Also note that we don't have the pay rate and hours variables anymore so we need to eliminate those from the print. Then refactor your printEmployee() method to simply System.out.println(toString());

Name: [firstName] [lastName]
ID: [employeeNum]
Department: [department]
Title: [jobTitle]

Abstract methods – We just need to define these so that they are required on the sub classes.

Part 2: Code Hourly Employee

Create a new java file called **HourlyEmployee.java**

Reference the following UML Diagram for what to code:

final HourlyEmployee extends Employee
-wage : double -hoursWorked : double
<<constructor>>HourlyEmployee(String fn, String ln, int en, String dept, String job, double w) +increaseHours() : void +increaseHours(double h) : void +calculateWeeklyPay() : double +annualRaise() : void +holidayBonus() : double +resetWeek() : void +setPay(double pay) : void +toString() : String

Notes on Methods

Constructor - The constructor accepts all that an employee requires as well as a double for wage, and sets hoursWorked to 0.0.

toString - Overrides the Employee toString, but calls the super.toString and adding "Wage: " + wage and "Hours Worked: " + hoursWorked at the end each on their new line. (This is basically what we had setup for last iteration of the Employee class). **Use the currency NumberFormat object on the wage variable.**

Name: [firstName] [lastName]
ID: [employeeNum]
Department: [department]
Title: [jobTitle]
Wage: [wage]
Hours Worked: [hoursWorked]

increaseHours() - This method adds 1 hour to hoursWorked variable

increaseHours(double h) - this method adds the value sent(**h**) to hoursWorked variable. This should also check so that we don't add negative hours to the employee.

Note: this was our addHours() methods from before

resetWeek() - sets hoursWorked to 0.0

annualRaise() - give a wage increase of 5%

holidayBonus() return the amount of pay for working 40 hours (40 * wage)

calculateWeeklyPay() - Return amount earned in the week using wage and hoursWorked. However, any hours worked over 40 is considered overtime, so you need to calculate those extra hours as one and a half of the wage (1.5 * wage) * (hours - 40). You will probably need an if statement or do some math with modulus to calculate the hours over 40.

```
if hours > 40:
     pay = 40 * payrate + (1.5 * payrate) * (hours-40)
else:
     pay = payrate * hours
```

setPay(double pay) – Should change the wage variable to the amount sent

Part 3: Code Salary Employee

Create a new java file called **SalaryEmployee.java**

Reference the following UML Diagram for what to code:

final SalaryEmployee extends Employee
-salary : double
<<constructor>>SalaryEmployee(String fn, String ln, int en, String dept, String job, double s) +calculateWeeklyPay() : double +annualRaise() : void +holidayBonus() : double +resetWeek() : void +setPay(double pay) : void +toString() : String

Notes on Methods:

Constructor - The constructor accepts all that an employee requires as well as a double for salary.

toString - Overrides the Employee toString, but calls the super.toString and adding "Salary: " + salary at the end of each on their own new line. **Use the NumberFormat currency object on the salary variable.**

Name: [firstName] [lastName]
ID: [employeeNum]
Department: [department]
Title: [jobTitle]
Salary: [salary]

calculateWeeklyPay() - returns the amount earned in the week by taking the salary and dividing it by 52 (the number of weeks in a year)

annualRaise() - Salary increased by 6.25%

holidayBonus() - Returns 3.365% of salary

resetWeek() - does nothing, but you must define this so leave it blank.

setPay(double pay) – Should change the salary variable to the amount sent

Part 4: Code Commission Employee

Create a new java file called **CommissionEmployee.java**

Reference the following UML Diagram for what to code:

final CommissionEmployee extends Employee
-sales : double -rate : double
<<constructor>>CommissionEmployee(String fn, String ln, int en, String dept, String job, double rate) +increaseSales() : void +increaseSales(double s) : void +calculateWeeklyPay() : double +annualRaise() : void +holidayBonus() : double +resetWeek() : void +setPay(double pay) : void +toString() : String

Notes on Methods:

Constructor - The constructor accepts all that an employee requires as well as a double for rate (in decimal form so 3.5% would be sent as .035). This also sets the sales to 0.0

toString - Overrides the Employee toString, but calls the super.toString, adding "Rate: " + rate and "Sales: " + sales each on a new line. **Use the percent and currency NumberFormat objects on rate and sales respectively.**

Name: [firstName] [lastName]
ID: [employeeNum]
Department: [department]
Title: [jobTitle]
Rate: [rate]
Sales: [sales]

calculateWeeklyPay() - Returns the rate * sales

annualRaise() - Rate percentage increased by .002. example if rates is 3.5% it would be .035 + .002 so .037 or 3.7%

holidayBonus() - no bonus, return 0.

resetWeek() - reset sales to 0.0

increaseSales() - adds 100 to sales

increaseSales(double s) - adds the value sent (**s**) to sales. Again we shouldn't have negative sales add so make sure that the value sent is positive before adding.

setPay(double pay) – Should change the rate variable to the amount sent

Part 5: Check Your Program

Running the EmployeeChecker should produce these EXACT results:

```
*** Initial Employee Printing ***
Name: Steve Rodgers
ID: 3781
Department: Sales
Title: Manager
Salary: $64,325.00

Name: Clint Barton
ID: 6847
Department: Sales
Title: Customer Representative
Rate: 2.65%
Sales: $0.00

Name: Tony Stark
ID: 5749
Department: Service
Title: Lead Service Manager
Wage: $32.85
Hours Worked: 0.0



*** Changed Employee sales/hours ***
Name: Steve Rodgers
ID: 3781
Department: Sales
Title: Manager
Salary: $64,325.00

Name: Clint Barton
ID: 6847
Department: Sales
Title: Customer Representative
Rate: 2.65%
Sales: $325.00

Name: Tony Stark
ID: 5749
Department: Service
Title: Lead Service Manager
Wage: $32.85
Hours Worked: 32.0


*** Calculate Weekly Pay ***
Steve Rodgers - Weekly Pay: $1,237.02
Clint Barton - Weekly Pay: $8.61
Tony Stark - Weekly Pay: $1,051.20
```

```
*** Reset the Week ***
Name: Steve Rodgers
ID: 3781
Department: Sales
Title: Manager
Salary: $64,325.00

Name: Clint Barton
ID: 6847
Department: Sales
Title: Customer Representative
Rate: 2.65%
Sales: $0.00

Name: Tony Stark
ID: 5749
Department: Service
Title: Lead Service Manager
Wage: $32.85
Hours Worked: 0.0



*** Annual Raises ***
Name: Steve Rodgers
ID: 3781
Department: Sales
Title: Manager
Salary: $68,345.31

Name: Clint Barton
ID: 6847
Department: Sales
Title: Customer Representative
Rate: 2.85%
Sales: $0.00

Name: Tony Stark
ID: 5749
Department: Service
Title: Lead Service Manager
Wage: $34.49
Hours Worked: 0.0
```

Submission

1. Compress the project folder and submit it to this assignment.

   to compress: On Windows, right click -> send to -> compressed .zip file

on Mac, right-click -> Compress

Hints/Tips (Before Submitting):

- Don't forget to have a header at the top of your file and include a Resource statement.
  - ***NOTE you need this for EACH java file***
- Use comments as needed. Include comments for methods and what is going on in each method.
- Follow all Java Styling Guides as covered
- Ask me questions early and often as needed, this is a little more challenging of a program to make.