Assignment

Overview

The first iteration of our Employee Manager(last module) was a great start. Though we are missing a few things that the Vacuum company needs. Mainly a way to hold Emergency Contact information for the employees. Currently with the system they have to keep it on a physical notecard in the employee's folder.

That is a little too much work and paper management for Gladys the office lady. So, we are going to make a new class that holds the contact information for one person, add an ArrayList into the Employee to hold contact information along with appropriate methods to use that ArrayList.

Directions

To Get Started:

1. Create a project in your IntelliJ (or your preferred IDE) titled **YOURNAME-Assignment8**
2. Download and import into your project this file: **A8EmployeeChecker.java** & **EmployeeSystem.java**
3. Take a copy of your Employee.java file from Assignment 7 and insert it into this project's src folder.
4. Again, look through the checker file and the updated system file. They have added functionality to have contacts in the employee class. These files will help you check your coding work on the Employee and new class EmergencyContact

Coding the file:

We need to do two things here. First create and Emergency Contact class that holds all the information that the company needs. Secondly modify the Employee class so that it has an ArrayList for the emergency contacts and methods to allow access.

Part 1: Emergency Contacts

Create a new java class called EmergencyContact.java. This will hold the name, relationship and phone number of an emergency contact of an employee.

Reference the following UML diagram:

EmergencyContact.java
-name : String -relationship : String -phone : String
<<constructor>>EmergencyContact(String n, String r, String p) +getName() : String +setName(String n) : void +getRelationship() : String +setRelationship(String r) : void +getPhone() : String +setPhone(String p) : void +printContact() : void

Notes on Methods/Variables:

Constructors

We will only have one constructor as we can't set up a blank contact. The constructor takes on 3 string (n – name, r – relationship, p – phone) that should get saved to the instance variables

Getters and Setters

These should be standard, they return and set values for each corresponding field

printContact()

This method should print out the information of the contact following this format:

Name: {name}
Relationship: {relationship}
Phone Number: {phone}

Part 2: Modify Employee with ArrayLists

Now that we have the EmergencyContact created we need a ArrayList on our employee to hold those. For this part we are going to add that along with some getter methods to help with our runner/checker files.

Reference the updated Employee UML diagram: (Changes are highlighted)

Employee.java
-firstName : String -lastName : String -employeeNum : int -department : String -jobTitle : String -hoursWorked : double -payRate : double -emergencyContacts : ArrayList<EmergencyContact> -currency : NumberFormat
<<constructor>>Employee(String fn, String ln, int en, String dept, String job, double pr) <<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 +getHoursWorked() : double +addHours() : void +addHours(double h) : void +resetHours() : void +getPayRate() : double +setPayRate(double pr) : void +calculateWeeklyPay() : double +printEmployee() : void +printEmergencyContacts() : void +clearContacts : void +addNewContact(EmergencyContact contact) : void +getEmergencyContactList() : ArrayList<EmergencyContact> +removeContact(EmergencyContact) : boolean +removeContact(int index) : boolean

Notes on Methods/Variables:

New Variable:

You will need to import the ArrayList and add a new ArrayList of type EmergencyContact into the employee. *NOTE: we are not initializing it, that is done in the constructors.*

Constructors

Now that we have an ArrayList we need to add in each constructor an initializing of it EXCEPT the copy constructor. In that constructor you should use the getEmergencyContactList() method. In the other constructors the ArrayList should be set to the new ArrayList<>(); code.

Getters

The method at the bottom of the UML diagram is our getter. It has a return type of the ArrayList and should return the `emergencyContact` variable.

printEmergencyContacts()

This method should loop through the ArrayList and print out each contact that is the list. But with a special format. Make sure to have a space between the title and between each contact. (see below)

First at the top BEFORE the loop you should print out:

****** Emergency Contacts ******

Then BEFORE printing each contact (so this would be in the for loop), you should print:

****** Contact # ****** ← Where the "#" is what contact we are looking at.

So the overall output would be something like:

```
**** Emergency Contacts ****

**** Contact 1 ****
Name: John Smith
Relationship: Friedn
Phone: 888-888-8888

**** Contact 2 ****
Name: Michelle Epps
Relationship: Spouse
Phone: 456-789-1236
```

clearContacts ()

This method should clear out the ArrayList

addNewContact(EmergencyContact contact)

This method will receive a new contact and add it into the emergency list.

removeContact(EmergencyContact contact)

This method will receive a contact to remove from the employee's list. Note that this method returns a Boolean data type. It should return true if the contact is in the list and if the removing was successful, otherwise it will return false.

To do this you should use an if statement and the ArrayList .contains() method. Inside you should use the .remove() method to remove from the list and return true. Use the else to return false.

removeContact(int index)

This another overloaded that method will remove a contact based on an index location along with return true/false if it was done successful.

For this though, we don't want to throw an exception SO you should also have an if statement making sure the size of the emergencyContact lists is >= index && > 0. This would be where you call the remove method to remove the index and return true, otherwise you return false.

Output of A8EmployeeChecker.java

```
*****  Assignment 8 Employee Checker *****

   ** Steve Rodgers **
**** Emergency Contacts ****

**** Contact 1****
Name: Peggy Carter
Relationship: Friend
Phone Number: 202-896-3633

**** Contact 2****
Name: Sharon Carter
```

```
Relationship: Friend
Phone Number: 202-836-9963

**** Contact 3****
Name: Nick Furry
Relationship: Boss
Phone Number: He will contact you


   ** Clint Barton **
**** Emergency Contacts ****

**** Contact 1****
Name: Laura Barton
Relationship: Spouse
Phone Number: 319-225-7451

**** Contact 2****
Name: Nick Furry
Relationship: Boss
Phone Number: He will contact you


   ** Tony Stark **
**** Emergency Contacts ****

**** Contact 1****
Name: Virginia Potts
Relationship: Spouse
Phone Number: 646-746-1025

**** Contact 2****
Name: Happy Hogan
Relationship: Bodyguard
Phone Number: 646-021-4411

**** Contact 3****
Name: Nick Furry
Relationship: Boss
Phone Number: He will contact you


   **Updated Tony Stark **
**** Emergency Contacts ****

**** Contact 1****
Name: Pepper Potts
Relationship: Spouse
Phone Number: 646-746-1025

**** Contact 2****
Name: Happy Hogan
Relationship: Bodyguard
Phone Number: 646-021-4411
```

```
**** Contact 3****
Name: Nick Furry
Relationship: Boss
Phone Number: He will contact you


   **Updated Steve Rodgers **
**** Emergency Contacts ****

**** Contact 1****
Name: Peggy Carter
Relationship: Friend
Phone Number: 202-896-3633

**** Contact 2****
Name: Nick Furry
Relationship: Boss
Phone Number: He will contact you


   **Updated (Again) Steve Rodgers **
**** Emergency Contacts ****

**** Contact 1****
Name: Peggy Carter
Relationship: Friend
Phone Number: 202-896-3633
```

Submission

1. Compress the IntelliJ 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.*
- *Use comments as needed. Include comments for methods and what is going on in each method.*
- *Follow all Java Styling Guides as covered*
- *Your output SHOULD MATCH the A8EmployeeChecker output exactly. Checking the printing of new lines using either \n OR a blank println() method.*
- *Ask me questions early and often as needed, this is a little more challenging of a program to make.*