

Organization Payment

Problem:

An organization has employees of two types: Salaried employees and contract employees. Each employee has a name, mailing address, email, and pay. Pay of a salaried employee is specified at the time of hire as amount per annum to be paid as 26 equal payments. The pay of a contract employees is specified each time when the contract is offered and is paid in three equal payments.

Develop a Java application that can prepare payment statement for all employees.

The output of the application should be:

| Name | Employee Type | Email | Payment amount |
|-------|---------------|-------|----------------|
| | | | \$dddd.dd |
| | | | \$dddd.dd |
| | | | \$dddd.dd |

Architecture of the project:

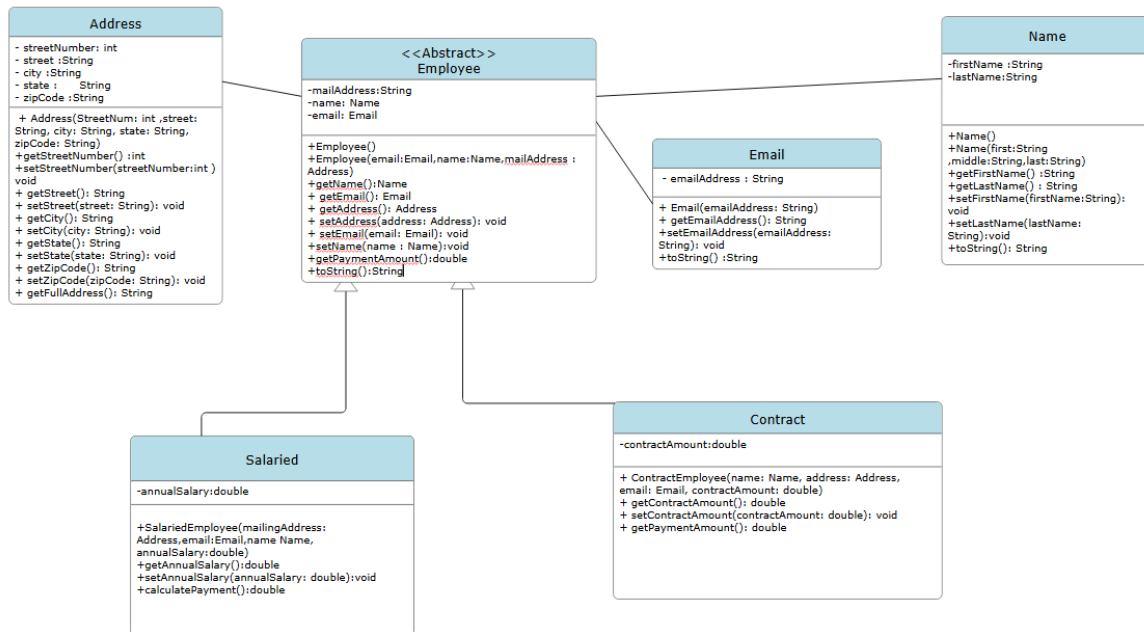
Make the mailing address as an instance of Address object/class.

Make the name variable as an instance of Name object/class.

Make the email variable as an instance of Email object/class.

The Employee is an abstract superclass with Contract and Salaried Employee as its subclass.

UML Diagram for this project:



Algorithm:

For Name, Email, and Address Class:

- 1.) Make a class for its address and all the attributes provided in UML diagram
- 2.) Define the constructors in the class
- 3.) define mutator and accessor for each of the attributes and provide a overridden toString method.

Employee Class:

- 1.) Define the class as an abstract class.
- 2.) Make the setters and getters for its attributes but use the corresponding object class like the name class.
- 3.) Make the payment calculator method as an abstract that can be overridden by subclass

For Salaried and Contract Employee class:

- 1.) Define the class as an extension of Employee class.
- 2.) Use the super() annotation in order to access the superclass attributes.
- 3.) Use the overridden payment calculator method of its superclass to calculate the payment of each employee type.

Main Class

- 1.) Make an application as the main class of the project.

- 2.) Make an array list with Employee as its element object.
- 3.) Make an instance object of each employee attribute such as name, email, address since they are all a separate object instead of being a primitive data type.
- 4.) Add all the instantiated variables as the argument for the Employee object.
- 5.) Make a separate method to generate the pay statement and its display.
- 6.) In the pay statement method, invoke the date built-in class from JVM in order to format the date in the statement.
- 7.) Identify if the employee type is contractual or salaried.
- 8.) Use an enhanced for loop to print out the name, employee type, email and their salary

Code:

Address sclass

```
/**
 * Name: Ralph Sabio
 * ID:991646078
 * Assigment 3
 * This is the Address class
 *
 *
 */

public class Address {
    private int streetNumber;
    private String street;
    private String city;
    private String state;
    private String zipCode;

    // arg constructor
    public Address(int streetNumber,String street, String city, String state,
String zipCode) {
        this.streetNumber=streetNumber;
        this.street = street;
        this.city = city;
        this.state = state;
        this.zipCode = zipCode;
    }

    // Getter and Setter methods for the attributes
    public String getStreet() {
```

```

        return this.street;
    }
    public int getStreetNumber(){
        return this.streetNumber;
    }
    public void setStreetNumber(int streetNumber){
        this.streetNumber = streetNumber;
    }
    public void setStreet(String street) {
        this.street = street;
    }

    public String getCity() {
        return this.city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getState() {
        return this.state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public String getZipCode() {
        return this.zipCode;
    }

    public void setZipCode(String zipCode) {
        this.zipCode = zipCode;
    }

    // Full address as a string in a toString method
    @Override
    public String toString() {
        return this.streetNumber+", "+this.street + ", " + this.city + ", " +
this.state + " " + this.zipCode;
    }
}

```

Contract Employee Class

```
// subclass for contract employee
class ContractEmployee extends Employee {
    private double contractAmount;

    // Constructor
    public ContractEmployee(Name name, Address address, Email email, double
contractAmount) {
        super(name, address, email); //inherited attributes from superclass
employee
        this.contractAmount = contractAmount;
    }

    // Getter and Setter methods
    public double getContractAmount() {
        return this.contractAmount;
    }

    public void setContractAmount(double contractAmount) {
        this.contractAmount = contractAmount;
    }

    // Override method to calculate payment amount
    @Override
    public double getPaymentAmount() {
        return this.contractAmount / 3; //three installment payment
    }
}
```

Email Class

```
public class Email {
    private String emailAddress;

    // arg constructor
    public Email(String emailAddress) {
        this.emailAddress = emailAddress;
    }

    // Getter and setter for email address
```

```

    public String getEmailAddress() {
        return this.emailAddress;
    }

    //
    public void setEmailAddress(String emailAddress) {
        this.emailAddress = emailAddress;
    }

    // toString method()
    @Override
    public String toString() {
        return this.emailAddress;
    }
}

```

Employee class

```

abstract class Employee {
    private Name name;
    private Address mailingAddress;
    private Email email;

    // Constructor
    public Employee(Name name, Address mailingAddress, Email email) {
        this.name = name;
        this.mailingAddress = mailingAddress;
        this.email = email;
    }

    // getters and setters for the name
    public Name getName() {
        return this.name;
    }

    public Address getMailingAddress() {
        return this.mailingAddress;
    }

    public void setAddress(Address mailingAddress) {
        this.mailingAddress = mailingAddress;
    }
}

```

```

    public Email getEmail() {
        return this.email;
    }

    public void setEmail(Email email) {
        this.email = email;
    }

    // Abstract method to calculate payment amount

    public abstract double getPaymentAmount();
    @Override
    public String toString() {
        return String.format("%s\t%s\t%s\t$%.2f",
name.toString(), email.toString(),mailingAddress.toString(),getPaymentAmount());
    }
}

```

Name class

```

public class Name {
    private String firstName;
    private String lastName;

    // Constructor with arguments
    public Name(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
    }

    // Getter and Setter methods
    public String getFirstName() {
        return this.firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {

```

```

        return this.lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    // @Override
    public String toString() {
        return this.firstName + " " + this.lastName;
    }
}

```

```

//class for the salary employee subclass
class SalariedEmployee extends Employee {
    private double annualSalary;

    // Constructor with arguments
    public SalariedEmployee(Name name, Address address, Email email, double
annualSalary) {
        super(name, address,email);//inherited from the superclass Employee
        this.annualSalary = annualSalary;
    }

    // Getter and Setter methods
    public double getAnnualSalary() {
        return this.annualSalary;
    }

    public void setAnnualSalary(double annualSalary) {
        this.annualSalary = annualSalary;
    }

    // Override method to calculate payment amount
    @Override
    public double getPaymentAmount() {
        return this.annualSalary / 26;// annual salary for 26 equal payment
    }
}

```


Main Class Application

```
/**
 * Name: Ralph Sabio
 * Date: March 27 2025
 * Application for the employee program
 */
import java.util.ArrayList;

import java.text.SimpleDateFormat;
import java.util.Date;

public class App {
    public static void main(String[] args) {
        // create an arrayList with Employee as an element object
        ArrayList<Employee> employees = new ArrayList<>();

        // make an object of name,address,and email
        Address address1 = new Address(45," Baker Street", "NYC", "New York",
"90845");
        Name name1 = new Name("Iroquois", "Pliskin");
        Email email1 = new Email("solSnake_32@fox.com");

        Address address2 = new Address(23," Kamurocho Street ", "Tokyo", "Kanto",
"67B09");
        Name name2 = new Name("Kiryu", "Kazuma");
        Email email2 = new Email("kkaz@gmail.com");

        Address address3 = new Address(78," Beverly Hills ", "Los Angeles",
"California", "90210");
        Name name3 = new Name("Travis", "Scott");
        Email email3 = new Email("scottTy@yahoo.com");

        // Creating employees and its type
        employees.add(new SalariedEmployee(name1, address1, email1, 52000));
        employees.add(new SalariedEmployee(name3, address3, email3, 78000));
        employees.add(new ContractEmployee(name2, address2, email2, 15000));

        // Generating payment statement via the getPaymentStatement method
        generatePaymentStatement(employees);
    }

    public static void generatePaymentStatement(ArrayList<Employee> employees) {
```

```

        // Format and display current date using the built-in date object/class
        String currentDate = new SimpleDateFormat("MM-dd-yyyy").format(new
Date());
        System.out.println("Employees Payment Statement");
        System.out.println("Date: " + currentDate); // Displaying the date in
the header
        System.out.println();
        System.out.printf("%-20s %-15s %-30s %-15s\n", "Name", "Employee Type",
"Email", "Payment Amount");
        System.out.println("-----
-----");

        for (Employee employee : employees) { // enhanced for loop
            String employeeType;
            if (employee instanceof SalariedEmployee) { // if employee
//is an instance of salaried type, set it to salaried else
contract
                employeeType = "Salaried";
            } else {
                employeeType = "Contract";
            }
            System.out.printf("%-20s %-15s %-30s $%-15.2f\n",
                employee.getName().toString(),
                employeeType,
                employee.getEmail(),
                employee.getPaymentAmount());
        }
    }
}

```

Output:

```

Employees Payment Statement
Date: 03-27-2025

Name                Employee Type  Email                                Payment Amount
-----
Iroquois Pliskin    Salaried      solSnake_32@fox.com                 $2000.00
Kiryu Kazuma        Salaried      kkaz@gmail.com                     $3115.38
Travis Scott        Contract      scotTy@yahoo.com                   $6666.67
PS C:\Users\ralph\Desktop\Sheridan\sem2\JAVA 1\Assignment3ByRalphSablo\src> 

```