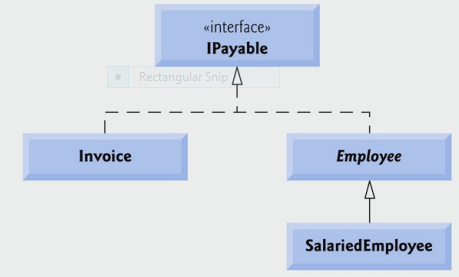
In Class Exercise

The UML class diagram shows the interface and class hierarchy used in an accounts­payable application.



**IPayable Interface**

Create an Interface called IPayable which contains the following:

**Data**

Low tax rate: 23%

High tax rate: 42%

VAT rate: 23.5%

**Method**

getPaymentAmount() which returns a double data type

**Invoice Class**

Create a class Invoice that represents a simple invoice that contains billing information for items that need to be purchased. This class implements the IPayable Interface

This class should have 4 member variables to represent the description, number, price and quantity of the item.

Create suitable getter and setter methods.

Override the toString() method.

Override the getPaymentAmount() method to return the amount that will need to be paid for this invoice (price \* quantity). VAT of 13.5% also needs to be applied to the invoice amount.

**Employee Class**

Create an abstract class Employee that represent employees who work in the company. This class implements the IPayable Interface

This class should have 3 member variables to represent the first name, last name and rsi number of an employee.

Create suitable getter and setter methods.

Override the toString() method.

**HourlyEmployee Class**

Create a class **HourlyEmployee** which is a sub class of Employee.

This class should have 2 additional member variables to represent hours worked and the rate of pay of an hourly employee.

Create suitable getter and setter methods.

Override the toString() method.

Override the getPaymentAmount() method to return the amount of wages that will need to be paid to the employee. Tax is charged on weekly wages as follows:

|  |  |
| --- | --- |
| **Range** | **Tax** |
| €0-€200 | No tax |
| >€200 and <= €300 | 23% |
| >€300 | Tax is 23% on the first €300  Tax is 42% on the remainder |

**Note**: the values for the ranges can also be defined in the Interface.

**Test Class**

Create a 5-element array to hold the following objects:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Invoice** | **Quantity** | **Price Per Item** | **Part Number** | **Description** |  |
|  | 2 | 375 | 01234 | Printer |  |
|  | 4 | 79.95 | 56789 | Ink Cartridges |  |
|  |  |  |  |  |  |
| **Employee** | **First Name** | **Last Name** | **RSI Number** | **Rate** | **Hours** |
|  | John | Smith | 6578431N | 30 | 60 |
|  | Lisa | Kelly | 9865431M | 12 | 20 |
|  | Mary | Reilly | 9875431M | 8 | 20 |

Call on the relevant methods to display information about each object. Use the sample output shown below as a guide.

In the sample output we also want to see the total amount of tax paid, the total amount of VAT paid, the total number of invoices processed and the total number of employees processed. Create static variable and methods in the relevant classes to achieve this.

**Sample Output:**

Invoice

Quantity=2

PricePerItem=375.0

PartNumber='01234'

PartDescription='Printer'

Amount Due: € 926.25

Invoice

Quantity=4

PricePerItem=79.95

PartNumber='56789'

PartDescription='Ink Cartridges'

Amount Due: € 394.953

Employee:

First Name='John'

Last Name='Smith'

RSI Number='6578431N'

John Smith is a salaried employee

WeeklySalary: € 1101.0

Employee:

First Name='Lisa'

Last Name='Kelly'

RSI Number='9865431M'

Lisa Kelly is a salaried employee

WeeklySalary: € 184.8

Employee:

First Name='Mary'

Last Name='Reilly'

RSI Number='9875431M'

Mary Reilly is a salaried employee

WeeklySalary: € 160.0

Write suitable Java code to calculate and display the summary information shown on the left hand side.

The total number of employees processed is: 3

The total number of invoices processed is: 2

The total amount of VAT paid is: €251.403

The total amount of tax paid is: €754.2