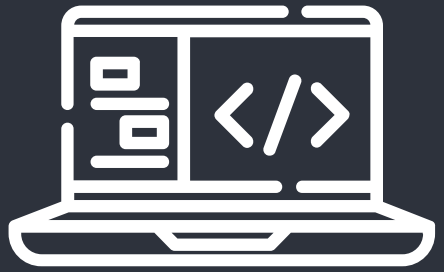

Paradigmas de Programación
Clase 6
1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1

</ Clase 6 - POO Integradora

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</>

Ejercicio



} /> [

1 0 1 1 0 1 1 0 1 1 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1

</Enunciado

La clase Empresa tiene una variable de instancia llamada empleados que es una colección que contiene todos los empleados de la empresa.

Cuando un empleado es de planta permanente, cobra la cantidad de horas trabajadas por $300 más 10 % por año de antigüedad más salario familiar. Cuando es de planta temporaria, tiene un contrato, no cobra antigüedad y cobra la cantidad de horas trabajadas por $200 más salario familiar. El salario familiar es $200 por cada hijo, los empleados casados además cobran $1000 por su esposa/o.

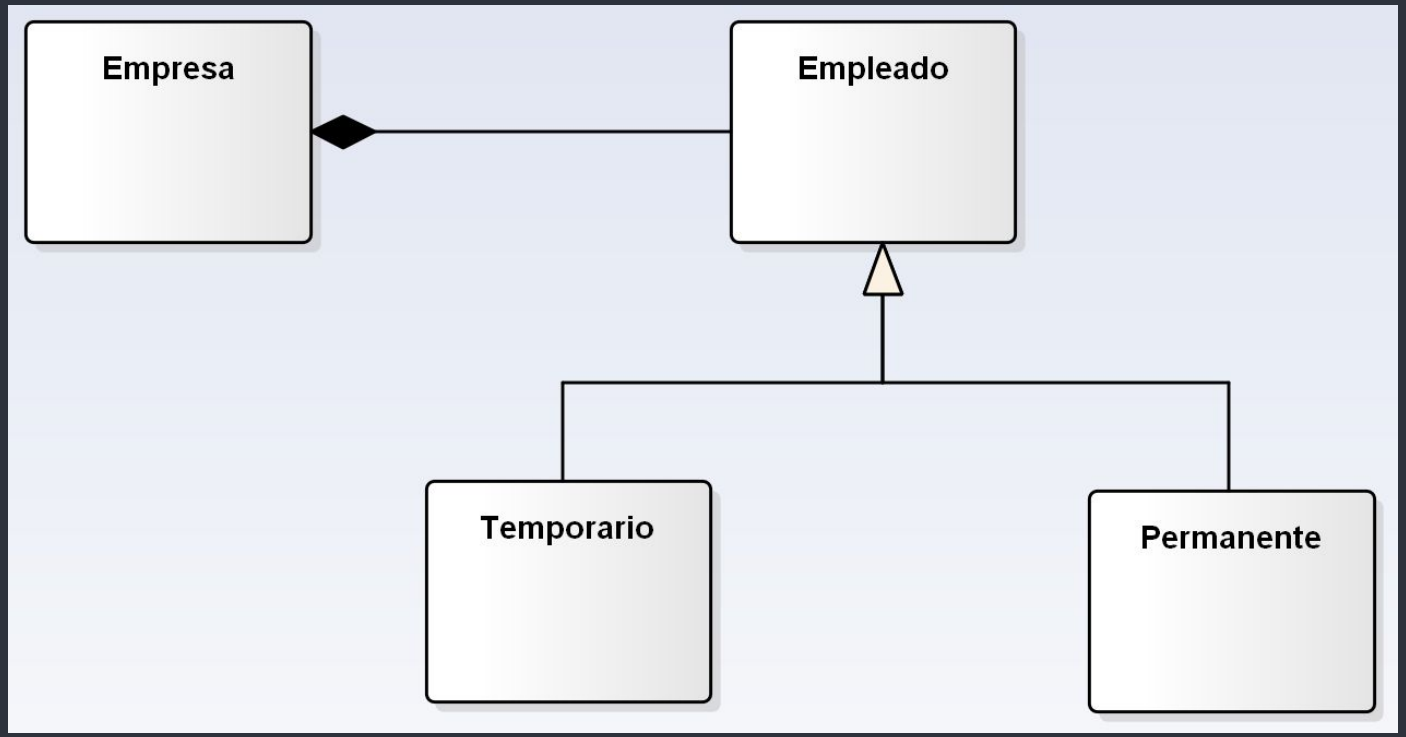
Implementar en aplicación el monto que la empresa debe pagar en concepto de sueldos a sus empleados que están de licencia.

Implementar en aplicación el monto que la empresa debe pagar en concepto de sueldos a sus empleados temporarios.

</Enunciado

Se desea conocer el monto que la empresa debe destinar a salarios familiares.
Además, el sistema debe permitir generar un listado de empleados con los montos que se le debe pagar en base a su antigüedad.
Generar un diccionario que muestre la cantidad de empleados por tipo (Temporario y Permanente)

</Diagrama de Clases



</Empresa: Definición y Métodos de clase

Implementacion Clase Empresa

Object subclass: **#Empresa**
instanceVariableNames: 'nombre empleados'
classVariableNames: ''
poolDictionaries: ''

Empresa class methods

>>crearEmpresa:unNom
“Crea la clase Empresa”
^(super new) initEmpresa:unNom

</Empresa: métodos de instancia

>>**initEmpresa:unNom**
“Inicializa la clase Empresa”
nombre:=unNom.
empleados:=OrderedCollection new.

>>modNombre:unNom
"Modifica el nombre de la Empresa por unNom"
nombre:=unNom.

>>verNombre "Retorna el nombre de la Empresa"
^nombre.

>>verEmpleados “Retorna todos los empleados de la empresa”
^empleados

</Empresa: métodos de instancia

```
>>verTotalEmpleados
“Retorna el total de empleados que tiene la Empresa”
^empleados size.

>>agregarEmpleado:emp
“Agrega un nuevo empleado emp a la Empresa”
 empleados add:emp.

>>eliminarEmpleado:emp
“Elimina un empleado emp de la Empresa”
 empleados remove:emp.
```

</Empresa: métodos de instancia

```
>>recuperarEmpleado:i
“Retorna el empleado de la posición i-esima”
^empleados at:i.

>>esVacia
“Valida si hay empleados en la empresa”
 ^empleados isEmpty.

>>existeEmpleado:emp
“Valida si está el empleado en la Empresa”
empleados includes:emp.
```

</Empleado: definición y métodos de clase

Object subclass: **#Empleado**
instanceVariableNames: 'nombre horas hijos licencia estcivil '
classVariableNames: ''
poolDictionaries: ''

Empleado class methods
>>crearEmp:unNom con:cantHr con:cantHi con:unaLic con:unEst
“Crea la clase Empleado”
^(super new) init:unNom con:cantHr con:cantHi con:unaLic con:unEst

</Empleado: métodos de instancia

```
>>init:unNom con:cantHr con:cantHi con:unaLic con:unEst
“Inicializa la clase Empleado”
nombre:=unNom.
horas:=cantHr.
hijos:=cantHi.
estcivil:=unEst.
licencia:=unEst.

>>modNom:unNom
"Modifica el nombre del empleado por unNom"
nombre:=unNom.
```

</Empleado: métodos de instancia

```
>>modCantH:unaCant
"Modifica la cantidad de horas del empleado por unaCant"
horas:=unaCant.

>>modHijos:unaCanth
"Modifica la cantidad de hijos del empleado por unaCanth'
hijos:=unaCanth.

>>modEstci:unEst
"Modifica el estado civil del empleado por unEst"
estcivil:=unEst.

>>modLic:unaLic
"Modifica la licencia del empleado por unaLic"
licencia:=unaLic.
```

</Empleado: métodos de instancia

```
>>verNom "Retorna el nombre del empleado"
^nombre.

>>verCantH "Retorna la cantidad de horas del empleado"
^horas.

>>verHijos "Retorna la cantidad de hijos del empleado"
^hijos.

>>verEstci "Retorna el estado civil del elemento del empleado"
^estcivil.

>>verLic "Retorna la licencia del empleado"
^licencia.
```

</Empleado: métodos de instancia

```
>>sueldo “Retorna el sueldo total del empleado”
^self básico + self salario

 >>básico “Retorna el sueldo básico del empleado (método abstracto)”
^self subclassResponsibility.

>>salario “Retorna el salario familiar del empleado”
| s |
s:= 200*hijos.
(estcivil = ‘Casado’) ifTrue: [s:= s + 1000].
^s.
```

</Permanente: Definición y métodos de clase

Empleado subclass: #Permanente
instanceVariableNames: 'antigüedad'
classVariableNames: ''
poolDictionaries: ''

Permanente class methods
>>crear:unNom con:ch con:cantH con:unaLicc on:unEst con:unaAnt

^(super new) init:unNom con:ch con:cantH con:unaLic con:unEst con:unaAnt.

</Permanente: Métodos de instancia

```
>>init:unNom con:ch con:cantH con:unaLic con:unEst con:unaAnt
“Inicializa la clase Permanente”
^super init:unNom con:ch con:cantH con:unaLic con:unEst.
antigüedad:=unaAnt.

>>modAntig:unaAnt “Modifica la antigüedad por unaAnt”
antigüedad:= unaAnt.

>>verAntig “Retorna la antigüedad del empleado”
^antigüedad.

>>básico “Retorna el sueldo básico del empleado permanente”
^ 300*horas + antigüedad * (300*horas*0,10).
```

</Temporario: Definición y métodos de clase

```
Empleado subclass: #Temporario
instanceVariableNames: 'contrato '
classVariableNames: ''
poolDictionaries:''

Temporario class methods

>>crear:unNom con:ch con:cantH con:unaLic con:unEst con:unCont
^super new init:unNom con:ch con:cantH con:unaLic con:unEst
con:unaCont.
```

</Temporario: Métodos de instancia

```
>>init:unNom con:ch con:cantH con:unaLic con:unEst con:unaAnt
con:unCont
“Inicializa la clase Temporario”
^super init:unNom con:ch con:cantH con:unaLic con:unEst.
contrato:=unCont.

>>modCont:unCont “Modifica el contrato por unCont”
contrato:= unCont.

>>verCont “Retorna el contrato del empleado”
^contrato.

>>básico “Retorna el sueldo básico del empelado temporario”
^200*horas.
```

</Enunciado