TITULO DE LABORATORIO

PARCIAL (SALARIO DE UN TRABAJADOR)

HANNA KATHERINE ABRIL GÓNGORA
KAROL ASLEY ORJUELA MAPE

UNIVERSIDAD MANUELA BELTRÁN

PROGRAMACIÓN ORIENTADA A OBJECTOS

DOCENTE

DIANA MARCELA TOQUICA RODRÍGUEZ

BOGOTÁ DC VIERNES 22 DE MARZO

1. **EXPLICACIÓN DEL CÓDIGO EN PYTHON**

- Se define una clase llamada Trabajador que tiene un método constructor *__init__*. Este método inicializa los atributos de un trabajador, como el nombre, documento, salario base, descuentos en porcentaje total, los cuales incluyen aporte a salud y aporte a pensión, y auxilio de transporte.

```
class Trabajador:
    def __init__(self, nombre, documento, salario_base, descuentos, aux_transporte):
        self.nombre = nombre
        self.documento = documento
        self.salario_base = salario_base
        self.descuentos = descuentos  # 4% salud + 4% pension
        self.aux_transporte = aux_transporte  # 162000.0

```

- El método *calcular_salario_total* calcula el salario total de un trabajador teniendo en cuenta todas las prebendas. Toma como parámetros las horas extras trabajadas y el valor por hora extra definido según el tipo de hora extra trabajada. Calcula el salario total sumando el salario base, aplicando un incremento del 10% por ley, agregando el valor de las horas extras y restando los descuentos de salud y pensión (calculados como un porcentaje del salario total). Finalmente, agrega el auxilio de transporte si el salario base es menor o igual a 2,000,000.

  Finalmente retorna el salario total, con horas extras y el total en descuentos.

```
    def calcular_salario_total(self, horas_extras, valor_hora_extra):
        salario_total = self.salario_base

        # Incremento del 10% por ley
        salario_total += salario_total * 0.1

        # Calculo horas extras
        salario_extra = horas_extras * valor_hora_extra
        salario_total += salario_extra

        # Calculo descuentos
        descuentos_total = salario_total * self.descuentos
        salario_total -= descuentos_total

        # Agregar auxilio de transporte
        if self.salario_base <= 2000000:
            salario_total += self.aux_transporte

        return salario_total, salario_extra, descuentos_total


```

- Se define una subclase *Horas_Extras* que hereda de la clase Trabajador. Esta subclase tiene un método *calcular_horas_extras* que calcula el salario extra basándose en el valor por hora extra y las horas extras trabajadas que ya están predefinidas en el código.

```
30  class Horas_Extras(Trabajador):
31      def calcular_horas_extras(self, valor_hora_extra, horas_extras):
32          salario_extra = horas_extras * valor_hora_extra
33          return salario_extra
34
```

- Se define una clase llamada *Registro_Trabajadores* que tiene una lista para almacenar trabajadores con información como el nombre y el documento de estos además tiene métodos para agregar trabajadores a la lista y listar los trabajadores existentes.

```
36  class Registro_Trabajadores:
37      def __init__(self):
38          self.trabajadores = []
39
40      def agregar_trabajador(self, trabajador):
41          self.trabajadores.append(trabajador)
42
43      def listar_trabajadores(self):
44          for trabajador in self.trabajadores:
45              print(f"Nombre: {trabajador.nombre}, Documento: {trabajador.documento}")
46
```

- Se define la función *main()* que crea un objeto de la clase *Registro_Trabajadores*. Luego, entra en un bucle donde solicita al usuario ingresar los datos de un trabajador, como nombre, documento, salario base y horas extras trabajadas.

```
48  def main():
49      registro = Registro_Trabajadores()
50
51      while True:
52          nombre = input("Ingrese el nombre del trabajador (o 'exit' para salir): ")
53          if nombre.lower() == 'exit':
54              break
55          documento = input("Ingrese el documento del trabajador: ")
56          salario_base = float(input("Ingresa el salario base: "))
57          horas_extras = int(input("Ingrese la cantidad de horas extras (Tener en cuenta que solo se toman horas enteras): "))
58
```

- Si las horas extras son menores o iguales a cero, el programa asume que el trabajador no tiene horas extras y calcula el salario total sin considerar horas extras y después imprime el comprobante de pago con dichos cálculos realizados

y después agrega el trabajador al registro.

```
59          if horas_extras ⩽ 0:
60              print("No tienes horas extras")
61              salario_total, salario_extra, descuentos_total = Trabajador(nombre, documento, salario_base, 0.08, 162000.0).calcular_salario_total(0, 0)
62
63              print("\n----- Comprobante de Pago Nomina -----")
64              print("_____________________________________")
65              print("\n------- Informacion Trabajador -------")
66              print("_____________________________________")
67              print(f"Nombre          |  {nombre}")
68              print(f"Documento       |  {documento}")
69              print("_____________________________________")
70              print(f"Horas Extras    |  {horas_extras}")
71              print(f"Salario Base    |  {salario_base}")
72              print(f"Salario Extra   |  {salario_extra}")
73              print(f"Descuentos      |  {descuentos_total}")
74              print("_____________________________________")
75              print(f"Salario Total   |  {salario_total}")
76              print("_____________________________________")
77
78              registro.agregar_trabajador(Trabajador(nombre, documento, salario_base, 0.08, 162000.0))
79
```

- En caso de que las horas extras sean mayores a cero, el programa entra en un bloque de código para manejar las horas extras y calcular el salario total teniendo en cuenta el tipo de horas extras trabajadas, además se maneja las diferentes opciones de horas extras, calcula el salario total, imprime el comprobante de pago

```
81          else:
82
83              print("""Tipo de horas extras trabajadas
84                        1. Hora extra diurna
85                        2. Hora extra nocturna
86                        3. Hora extra dominical festivos (diurna)
87                        4. Hora extra dominical festivos (nocturna)""")
88              op_extras = int(input("Ingresa el tipo de horas extras trabajadas: "))
89
90              trabajador = Horas_Extras(nombre, documento, salario_base, 0.08, 162000.0)
91
92              if op_extras == 1:
93                  salario_total, salario_extra, descuentos_total = trabajador.calcular_salario_total(horas_extras, 6915.0)
94              elif op_extras == 2:
95                  salario_total, salario_extra, descuentos_total = trabajador.calcular_salario_total(horas_extras, 9681.0)
96              elif op_extras == 3:
97                  salario_total, salario_extra, descuentos_total = trabajador.calcular_salario_total(horas_extras, 11064.0)
98              elif op_extras == 4:
99                  salario_total, salario_extra, descuentos_total = trabajador.calcular_salario_total(horas_extras, 13830.0)
100             else:
101                 print("Opción de horas extras no válida")
102                 return
103
104             registro.agregar_trabajador(trabajador)
105
106             print("\n----- Comprobante de Pago Nomina -----")
107             print("_____________________________________")
108             print("\n------- Informacion Trabajador -------")
109             print("_____________________________________")
110             print(f"Nombre          |  {trabajador.nombre}")
111             print(f"Documento       |  {trabajador.documento}")
112             print("_____________________________________")
113             print(f"Horas Extras    |  {horas_extras}")
114             print(f"Salario Base    |  {salario_base}")
115             print(f"Salario Extra   |  {salario_extra}")
116             print(f"Descuentos      |  {descuentos_total}")
117             print("_____________________________________")
118             print(f"Salario Total   |  {salario_total}")
119             print("_____________________________________")
120
```

- El código restante en la función *main()* ofrece la opción de listar todos los trabajadores registrados.

```
122             listar_todos = input("¿Deseas listar todos los trabajadores? (s/n): ")
123             if listar_todos.lower() == 's':
124                 print("\n----- Lista de Trabajadores -----")
125                 registro.listar_trabajadores()
126             if listar_todos.lower() == 'n':
127                 print("Saliento del sistema...")
128                 break
```

- Finalmente, se ejecuta la función *main()* si el script se ejecuta como el programa principal.

```
134 if __name__ == '__main__':
135     main()
136
```

2. **EXPLICACION DEL CODIGO UML**

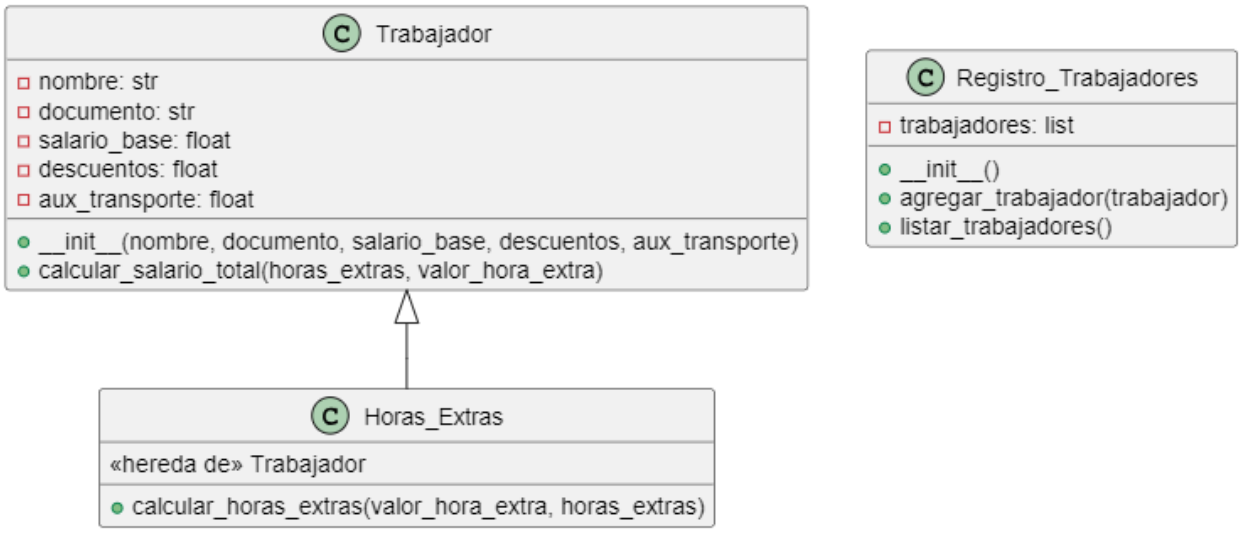
```
1   @startuml Salario_Trabajador
2
3   class Trabajador {
4     - nombre: str
5     - documento: str
6     - salario_base: float
7     - descuentos: float
8     - aux_transporte: float
9     + __init__(nombre, documento, salario_base, descuentos, aux_transporte)
10    + calcular_salario_total(horas_extras, valor_hora_extra)
11  }
12
13  class Horas_Extras {
14    <<hereda de>> Trabajador
15    + calcular_horas_extras(valor_hora_extra, horas_extras)
16  }
17
18  class Registro_Trabajadores {
19    - trabajadores: list
20    + __init__()
21    + agregar_trabajador(trabajador)
22    + listar_trabajadores()
23  }
24
25  Trabajador <|-- Horas_Extras
26
27  @enduml
28
```

- **Líneas 1-11:** Se define una clase llamada “Trabajador” con los siguientes atributos:

- nombre: Representa el nombre del trabajador (tipo de dato: cadena de caracteres).
- documento: Representa el número de documento del trabajador (tipo de dato: cadena de caracteres).
- salario_base: Indica el salario base del trabajador (tipo de dato: número decimal).
- descuentos: Representa los descuentos aplicados al salario (tipo de dato: número decimal).
- aux_transporte: Es el valor del auxilio de transporte (tipo de dato: número decimal).
- El método _init_ (constructor) inicializa los atributos con los valores proporcionados.
- El método calcular_salario_total calcula el salario total teniendo en cuenta las horas extras trabajadas y el valor de la hora extra.

- **Líneas 13-15:** Se define una clase llamada "Horas_Extras" que hereda de la clase "Trabajador". Esta clase tiene un método:
  - calcular_horas_extras: Calcula las horas extras basado en el valor de la hora extra y las horas extras trabajadas.
  - Líneas 17-22: Se define una clase llamada "Registro_Trabajadores" con un atributo:
  - lista_trabajadores: Es una lista que almacena objetos de la clase "Trabajador".
  - Los métodos agregar_trabajador y listar_trabajadores permiten agregar nuevos trabajadores a la lista y listar los existentes.
- **Línea 24:** Indica la relación de herencia entre las clases "Trabajador" y "Horas_Extras".
- **Línea 27:** Marca el final del diagrama UML.

3. **DIAGRAMA DEL CODIGO EN UML**



4. **EXPLICACION Y USO**

El código proporciona un sistema de registro y cálculo de nómina para trabajadores. Se define una clase llamada Trabajador que tiene atributos como nombre, documento, salario base, descuentos por salud y pensión, y auxilio de transporte. La clase también tiene un método para calcular el salario total, considerando horas extras, incrementos legales, descuentos de salud y pensión además de auxilio de transporte. Además, existe una clase Horas_Extras que hereda de Trabajador y permite calcular el salario extra por horas extras trabajadas. La clase Registro_Trabajadores administra una lista de trabajadores y proporciona métodos para agregar trabajadores al registro y listarlos.

La función principal (main) del programa interactúa con el usuario para ingresar información de los trabajadores, como nombre, documento, salario base y horas extras. Dependiendo de si hay horas extras o no, se calcula el salario total y se genera un comprobante de pago con los detalles del trabajador y su salario. También se ofrece la opción de listar todos los trabajadores registrados en el sistema.

En resumen, este código simple facilita la gestión de información y el cálculo de salarios para empleados, brindando una herramienta útil para la administración de nóminas.

5. DEMOSTRACION DE PRUEBAS Y FUNCIONAMIENTO DEL CODIGO

5.1. Prueba 1.

```
Ingrese el nombre del trabajador (o 'exit' para salir): Hanna Abril
Ingrese el documento del trabajador: 1006258532
Ingresa el salario base: 2700000
Ingrese la cantidad de horas extras (Tener en cuenta que solo se toman horas enteras): 5
Tipo de horas extras trabajadas
                    1. Hora extra diurna
                    2. Hora extra nocturna
                    3. Hora extra dominical festivos (diurna)
                    4. Hora extra dominical festivos (nocturna)
Ingresa el tipo de horas extras trabajadas: 1

----- Comprobante de Pago Nomina -----
----------------------------------------

------- Informacion Trabajador -------
----------------------------------------
Nombre          |  Hanna Abril
Documento       |  1006258532
----------------------------------------
Horas Extras    |  5
Salario Base    |  2700000.0
Salario Extra   |  34575.0
Descuentos      |  240366.0
----------------------------------------
Salario Total   |  2764209.0
----------------------------------------
¿Deseas listar todos los trabajadores? (s/n): s

----- Lista de Trabajadores -----
Nombre: Hanna Abril, Documento: 1006258532
```

En esta primera prueba se demuestra el funcionamiento con el calculo del salario de un trabajador, en donde se le piden que ingrese la información de esta y que escoja el tipo de horas extras, después de eso se le da el respectivo comprobante de nomina al usuario con toda la información, además de eso se muestra que se agrego el nombre del trabajador a la lista de control.

5.2. Prueba 2.

```
Ingrese el nombre del trabajador (o 'exit' para salir): Daniel Juarez
Ingrese el documento del trabajador: 159357
Ingresa el salario base: 1200000
Ingrese la cantidad de horas extras (Tener en cuenta que solo se toman horas enteras): 10
Tipo de horas extras trabajadas
                    1. Hora extra diurna
                    2. Hora extra nocturna
                    3. Hora extra dominical festivos (diurna)
                    4. Hora extra dominical festivos (nocturna)
Ingresa el tipo de horas extras trabajadas: 2

----- Comprobante de Pago Nomina -----
________________________________________

------- Informacion Trabajador -------
________________________________________
Nombre          |  Daniel Juarez
Documento       |  159357
________________________________________
Horas Extras    |  10
Salario Base    |  1200000.0
Salario Extra   |  96810.0
Descuentos      |  113344.8
________________________________________
Salario Total   |  1465465.2
________________________________________
¿Deseas listar todos los trabajadores? (s/n): s

----- Lista de Trabajadores -----
Nombre: Hanna Abril, Documento: 1006258532
Nombre: Daniel Juarez, Documento: 159357
```

En esta prueba se inscribe otro trabajador con su respectivo salario y sus horas extras en este caso se elije otro tipo de horas extras que tienen otro valor, y se hace el respectivo calculo para después pasar el comprobante de nomina al usuario, e igual que la anterior se ve listado este usuario y el anterior.

5.3. Prueba 3.

```
Ingrese el nombre del trabajador (o 'exit' para salir): Sarah Martinez
Ingrese el documento del trabajador: 123456
Ingresa el salario base: 2500000
Ingrese la cantidad de horas extras (Tener en cuenta que solo se toman horas enteras): 0
No tienes horas extras

----- Comprobante de Pago Nomina -----
________________________________________

------- Informacion Trabajador -------
________________________________________
Nombre          |  Sarah Martinez
Documento       |  123456
________________________________________
Horas Extras    |  0
Salario Base    |  2500000.0
Salario Extra   |  0
Descuentos      |  220000.0
________________________________________
Salario Total   |  2530000.0
________________________________________
```

En esta prueba podemos evidenciar que este trabajador inscrito no realizo horas extras, por lo que en el comprobante de pago se puede evidenciar, que no tiene salario extra y que además no hay horas extras.

5.4. Prueba 4.

```
Ingrese el nombre del trabajador (o 'exit' para salir): Laura Reyez
Ingrese el documento del trabajador: 1300000
Ingresa el salario base: 1500000
Ingrese la cantidad de horas extras (Tener en cuenta que solo se toman horas enteras): 1
Tipo de horas extras trabajadas
                    1. Hora extra diurna
                    2. Hora extra nocturna
                    3. Hora extra dominical festivos (diurna)
                    4. Hora extra dominical festivos (nocturna)
Ingresa el tipo de horas extras trabajadas: 4

----- Comprobante de Pago Nomina -----
----------------------------------------

------- Informacion Trabajador -------
----------------------------------------
Nombre          |  Laura Reyez
Documento       |  1300000
----------------------------------------
Horas Extras    |  1
Salario Base    |  1500000.0
Salario Extra   |  13830.0
Descuentos      |  133106.4
----------------------------------------
Salario Total   |  1692723.6
----------------------------------------
```

En esta prueba se evidencia el uso de otro tipo de hora extra, e igual que todas las anteriores retorna el comprobante de nómina con la información respectiva.

5.5. Prueba 5.

```
Ingrese el nombre del trabajador (o 'exit' para salir): Cisco Neira
Ingrese el documento del trabajador: 147852
Ingresa el salario base: 2300000
Ingrese la cantidad de horas extras (Tener en cuenta que solo se toman horas enteras): 3
Tipo de horas extras trabajadas
                    1. Hora extra diurna
                    2. Hora extra nocturna
                    3. Hora extra dominical festivos (diurna)
                    4. Hora extra dominical festivos (nocturna)
Ingresa el tipo de horas extras trabajadas: 3

----- Comprobante de Pago Nomina -----
----------------------------------------

------- Informacion Trabajador -------
----------------------------------------
Nombre          |  Cisco Neira
Documento       |  147852
----------------------------------------
Horas Extras    |  3
Salario Base    |  2300000.0
Salario Extra   |  33192.0
Descuentos      |  205055.36000000002
----------------------------------------
Salario Total   |  2358136.64
----------------------------------------
¿Deseas listar todos los trabajadores? (s/n): s

----- Lista de Trabajadores -----
Nombre: Hanna Abril, Documento: 1006258532
Nombre: Daniel Juarez, Documento: 159357
Nombre: Sarah Martinez, Documento: 123456
Nombre: Laura Reyez, Documento: 1300000
Nombre: Cisco Neira, Documento: 147852
Ingrese el nombre del trabajador (o 'exit' para salir): exit
```

En esta prueba final podemos ver que se inscribió otro usuario y se uso otro tipo de hora extra, e igual que los anteriores dio el comprobante de nómina, pero en este caso lo importante es que se puede evidenciar que se almaceno los nombres de todos los trabajadores anteriores al momento de listarlos y que además de eso el sistema de salida del proceso funciona perfectamente.

6. DECISIÓN DE DISEÑO O CONSIDERACION IMPORTANTE

La principal consideración importante para el diseño era saber que parámetros importantes se necesitaban para obtener el salario legal mensual vigente de un trabajador en Colombia, para eso nos referimos a MinTrabajo, para revisar todo el proceso tanto de el salario base como de descuentos legales, además de eso horas extras y auxilios como en este caso de transporte.

Otra de las decisiones importantes que se tomaron era como se iba a mostrar dicha información al usuario, por lo que se decidió por hacer una especie de comprobante de nomina con toda la información importante, incluyendo nombre y el documento del trabajador.

7. PILARES DE POO IMPLEMENTADOS

7.1. Encapsulamiento:

Se utiliza la encapsulación al definir la clase Trabajador con sus atributos privados (nombre, documento, salario_base, descuentos, aux_transporte) estos únicamente serán accedidos desde la misma clase.

7.2. Abstracción:

La abstracción se implementa al definir las clases Trabajador, Horas_Extras, y Registro_Trabajadores, donde cada una representa un concepto específico del mundo real y proporciona métodos para interactuar con esos conceptos de manera abstracta sin preocuparse por los detalles internos de implementación.

7.3. Herencia:

Se utiliza la herencia al definir la clase Horas_Extras como clase hija de Trabajador. La clase hija hereda los atributos y métodos de la superclase Trabajador, lo que permite reutilizar código.

8. CONCLUCIONES

Al realizar el código anterior, se aprendieron varios conceptos importantes relacionados con la programación en Python y el diseño de sistemas básicos de administración de datos y cálculos. Algunas de las conclusiones son:

- Programación Orientada a Objetos (POO): Se utilizó la POO para definir clases como Trabajador, Horas_Extras y Registro_Trabajadores, lo cual facilitó la organización y reutilización del código al encapsular atributos y métodos relacionados.
- Herencia y Polimorfismo: Se aplicó el concepto de herencia al crear la clase Horas_Extras que hereda de Trabajador, lo que permitió reutilizar funcionalidades y métodos. Además, se observó un nivel básico de polimorfismo al utilizar métodos comunes en diferentes clases.

- Interacción con el Usuario: Se implementó la entrada de datos mediante la función input() para interactuar con el usuario y obtener información como nombre, documento y horas extras. Esto es fundamental para crear programas interactivos y amigables para el usuario.
- Cálculos y Lógica de Negocio: Se realizó la lógica de cálculo de salarios teniendo en cuenta el salario base, horas extras, descuentos y otros factores, lo cual es esencial en aplicaciones de gestión de nóminas y finanzas.
- Estructuras de Datos: Se utilizó una lista para almacenar objetos de tipo Trabajador en la clase Registro_Trabajadores, lo que demostró cómo gestionar y manipular datos en estructuras de datos comunes en Python.

En resumen, al realizar este código se adquirieron habilidades en programación orientada a objetos, manejo de datos, interacción con el usuario y lógica de negocios, aspectos fundamentales para el desarrollo de aplicaciones más complejas y funcionales en Python y otros lenguajes de programación.