UEA
UNIVERSIDAD
ESTATAL AMAZÓNICA

UNIVERSIDAD ESTATAL AMAZÓNICA

INGENIERÍA EN TECNOLOGÍAS DE INFORMACIÓN

PROGRAMACIÓN ORIENTADA A OBJETOS

TEMA:

CLASES, OBJETOS, HERENCIA, ENCAPSULAMIENTO Y POLIMORFISMO

ESTUDIANTE:

EDWIN FABIÁN NORIEGA BALDEON

DOCENTE:

ING. SANTIAGO ISRAEL NOGALES GUERRERO

NIVEL:

SEGUNDO NIVEL_PARALELO “A”

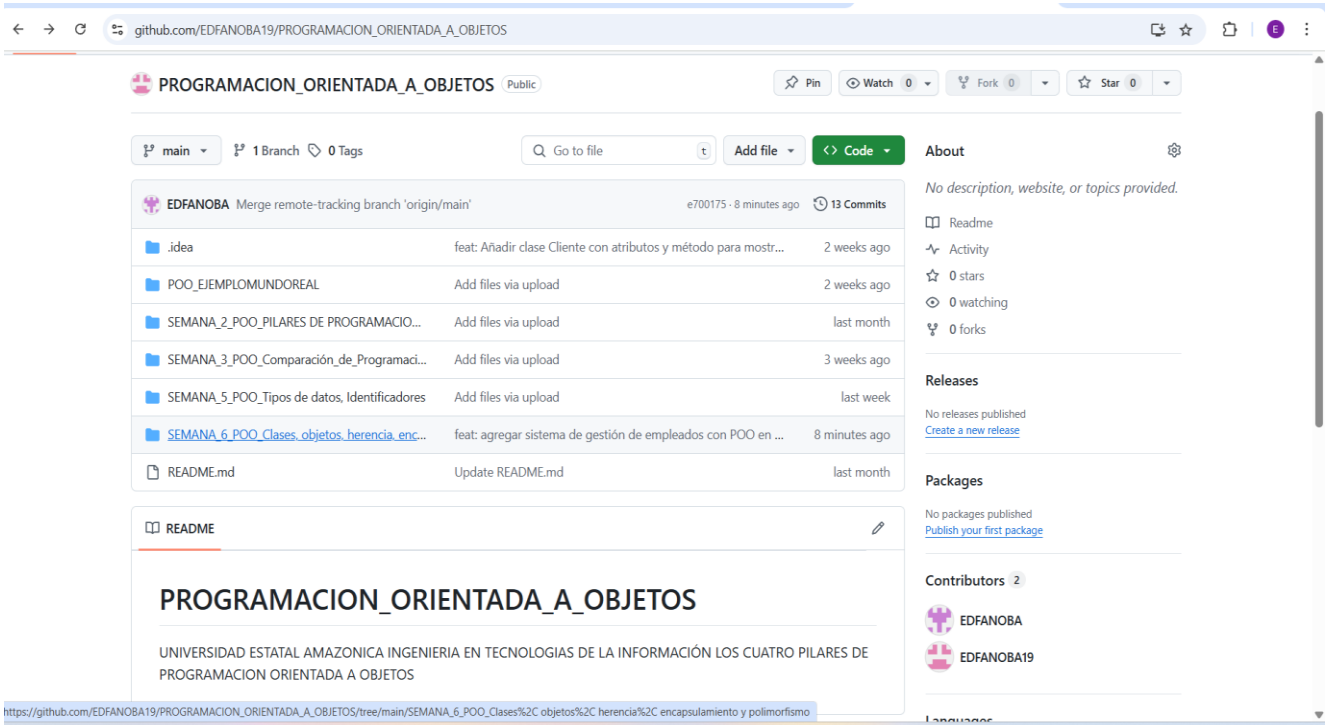
2025-2025

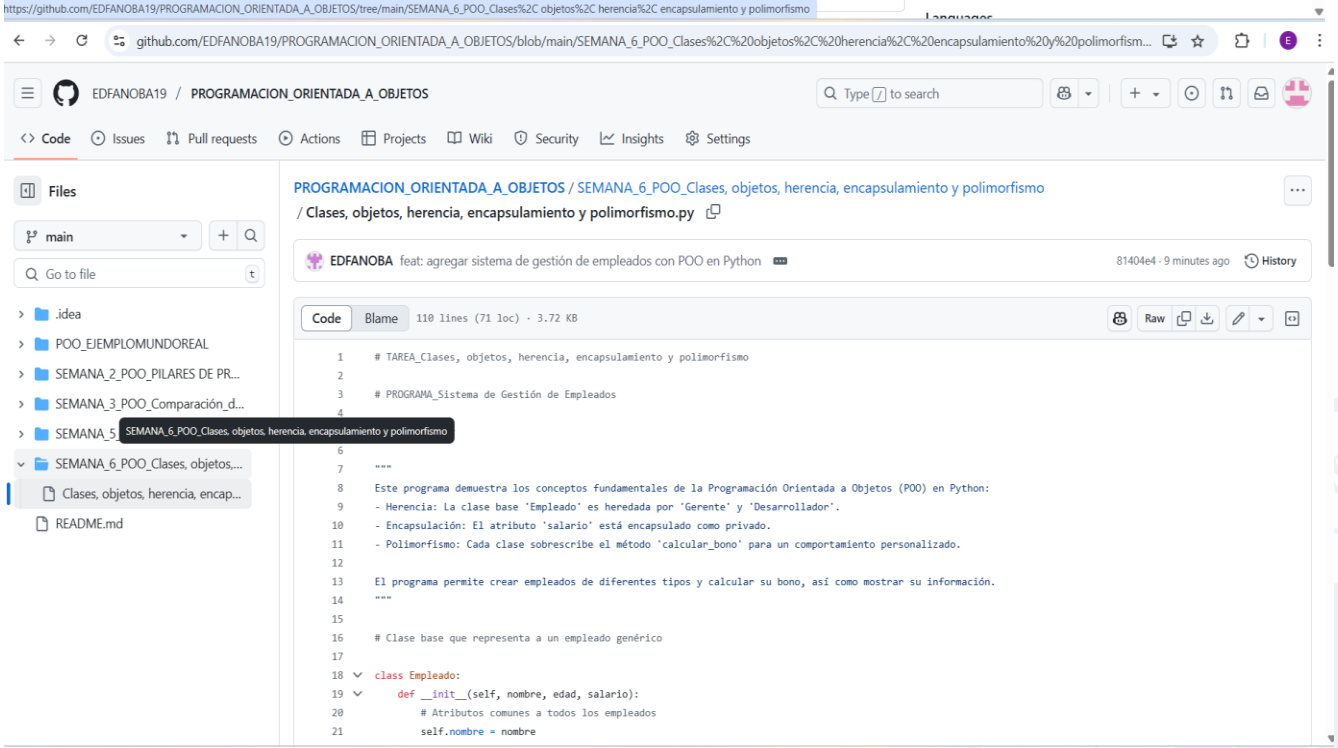
PUYO-ECUADOR

1. CÓDIGO DE GITHUB

https://github.com/EDFANOBA19/PROGRAMACION_ORIENTADA_A_OBJETOS.git

2. REPOSITORIO DE GITHUB





2.1. CLASES, OBJETOS, HERENCIA, ENCAPSULAMIENTO Y POLIMORFISMO

Programa Python: Sistema de gestión de empleados

Este programa fue desarrollado con el fin de aplicar todos los conceptos principales de POO, mediante la utilización de Python. Este programa simula un sistema simple de gestión de empleados dentro de una empresa, donde hay varios tipos de trabajadores con características y comportamientos propios.

El código contiene:

- **Una clase base** (empelado), que tiene atributos comunes como nombre, edad, y salario. El atributo (salario), esta encapsulado para demostrar control de accesos mediante métodos getter y setter.
- **Clase derivadas** (gerente) y (desarrollador), que heredan de (empleado), y agregan atributos específicos como departamento y lenguaje de programación respectivamente, las dos clases sobrescriben el método (calcular_bono), aplicando el polimorfismo.
- Una sección (if_name_==”_main_”:), donde se crean objetos de cada tipo, se muestra su información y se prueba la modificación del salario usando los métodos de encapsulamiento.

Conceptos de POO aplicados

- Herencia-Gerente y desarrollador heredan de la clase empleado
- Encapsulación-El atributo salario es privado, se accede solo por métodos
- Polimorfismo-Cada clase redefine el método calcular_bono() a su manera

2.1.1. CÓDIGO

PROGRAMACION_ORIENTADA_A_OBJETOS main Clases, objetos, herencia, encap...morfism

Clases, objetos, herencia, encapsulamiento y polimorfismo.py

```python
# TAREA_Clases, objetos, herencia, encapsulamiento y polimorfismo

# PROGRAMA_Sistema de Gestión de Empleados

# empleados.py

"""
Este programa demuestra los conceptos fundamentales de la Programación Orientada a Objetos (POO) en Python:
- Herencia: La clase base 'Empleado' es heredada por 'Gerente' y 'Desarrollador'.
- Encapsulación: El atributo 'salario' está encapsulado como privado.
- Polimorfismo: Cada clase sobrescribe el método 'calcular_bono' para un comportamiento personalizado.

El programa permite crear empleados de diferentes tipos y calcular su bono, así como mostrar su información.
"""

# Clase base que representa a un empleado genérico

class Empleado:  2 usages  new *
    def __init__(self, nombre, edad, salario):  new *
        # Atributos comunes a todos los empleados
        self.nombre = nombre
        self.edad = edad
        self.__salario = salario  # Atributo privado (encapsulado)

    # Método que puede ser sobrescrito por las clases hijas para calcular un bono

    def calcular_bono(self):  new *
        return self.__salario * 0.05  # Bono base del 5%

    # Getter: permite acceder al salario de forma controlada

    def obtener_salario(self):  4 usages  new *
        return self.__salario

    # Setter: permite modificar el salario, validando que no sea negativo

    def modificar_salario(self, nuevo_salario):  1 usage  new *
        if nuevo_salario >= 0:
            self.__salario = nuevo_salario
        else:
            print("El salario no puede ser negativo.")

    # Método para mostrar información general del empleado

    def mostrar_info(self):  2 usages  new *
        return f"Nombre: {self.nombre}, Edad: {self.edad}, Salario: {self.__salario}"

# Clase derivada que representa a un gerente

class Gerente(Empleado):  1 usage  new *
    def __init__(self, nombre, edad, salario, departamento):  new *
        # Llamamos al constructor de la clase base
        super().__init__(nombre, edad, salario)
        self.departamento = departamento  # Atributo adicional específico del gerente

    # Sobrescribimos el método para calcular un bono mayor para gerentes

    def calcular_bono(self):  1 usage  new *
        return self.obtener_salario() * 0.20  # Bono del 20%

    # Mostramos información extendida incluyendo el departamento
```

```python
    def mostrar_info(self):  1 usage  new *
        return super().mostrar_info() + f", Departamento: {self.departamento}"

# Clase derivada que representa a un desarrollador

class Desarrollador(Empleado):  1 usage  new *
    def __init__(self, nombre, edad, salario, lenguaje):  new *
        # Llamamos al constructor de la clase base
        super().__init__(nombre, edad, salario)
        self.lenguaje = lenguaje  # Atributo adicional específico del desarrollador

    # Sobrescribimos el método para calcular un bono distinto para desarrolladores

    def calcular_bono(self):  1 usage  new *
        return self.obtener_salario() * 0.10  # Bono del 10%

    # Mostramos información extendida incluyendo el lenguaje de programación

    def mostrar_info(self):  1 usage  new *
        return super().mostrar_info() + f", Lenguaje: {self.lenguaje}"


# Bloque principal del programa donde se crean objetos y se usan los métodos

if __name__ == "__main__":

    # Creamos un objeto de tipo Gerente

    gerente = Gerente( nombre: "Laura",  edad: 40,  salario: 5000,  departamento: "Marketing")

    # Creamos un objeto de tipo Desarrollador

    dev = Desarrollador( nombre: "Carlos",  edad: 28,  salario: 3000,  lenguaje: "Python")

    # Mostramos la información de cada empleado

    print(gerente.mostrar_info())
    print("Bono del gerente:", gerente.calcular_bono())

    print(dev.mostrar_info())
    print("Bono del desarrollador:", dev.calcular_bono())

    # Demostración de encapsulamiento: accedemos y modificamos el salario mediante métodos
    print("\nSalario original del desarrollador:", dev.obtener_salario())
    dev.modificar_salario(3500)  # Cambiamos el salario
    print("Nuevo salario del desarrollador:", dev.obtener_salario())
```

Run Clases, objetos, herencia, encapsulamiento y polimorfis...

```
"C:\Users\hp\PycharmProjects\SEMANA_2_POO_PILARES DE PROGRAMACION ORIENTADA A OBJETOS\.venv\Scripts\python.exe" "C:\Users\hp
Nombre: Laura, Edad: 40, Salario: 5000, Departamento: Marketing
Bono del gerente: 1000.0
Nombre: Carlos, Edad: 28, Salario: 3000, Lenguaje: Python
Bono del desarrollador: 300.0

Salario original del desarrollador: 3000
Nuevo salario del desarrollador: 3500

Process finished with exit code 0
```

N_ORIENTADA_A_OBJETOS › SEMANA_6_POO_Clases, objetos, herencia, encapsulamiento y polimorfismo › Clases, objetos, herencia, encapsulamiento y polimorfism