**UNIVERSIDAD MAYOR DE SAN ANDRES   
FACULTAD DE CIENCIAS PURAS Y NATURALES   
CARRERA DE INFORMATICA**

Logotipo

El contenido generado por IA puede ser incorrecto.

**PRACTICA NUMERO**

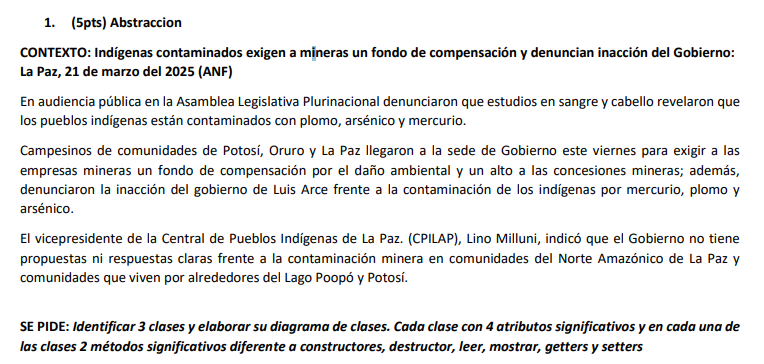
**10**

**NOMBRE:** EDWARD YAMIR LIZÁRRAGA ZABALA

**C.I.** 12479840

**LICENCIADO:** JHONNY FELIPEZ ANDRADE

PRIMER PARCIAL FILA A



Diagrama

El contenido generado por IA puede ser incorrecto.

Clase: Comunidad

* agregar\_contaminante()

Agrega un contaminante (por ejemplo plomo, arsénico) a la lista de contaminantes que

afectan a esa comunidad.

Ejemplo: agregar\_contaminante("plomo")

* calcular\_afectacion()

Calcula o estima el nivel de afectación en la comunidad según la cantidad y tipo de

contaminantes.

Ejemplo: calcula un índice de impacto ambiental.

Clase: EmpresaMinera

* pagar\_compensacion()

Simula o registra el pago de una compensación económica a las comunidades afectadas.

Ejemplo: pagar\_compensacion(100000 Bs.)

* aumentar\_extraccion()

Simula o registra el aumento de extracción de minerales.

Ejemplo: aumentar\_extraccion(500 toneladas/año)

Clase: Gobierno

* agregar\_accion()

Registra una acción que toma el gobierno frente al problema (como enviar inspección, crear fondo, etc.).

Ejemplo: agregar\_accion("inspección minera")

* aumentar\_fondo()

Aumenta el fondo de compensación total disponible para comunidades.

Ejemplo: aumentar\_fondo(500000 Bs.)

Tabla

El contenido generado por IA puede ser incorrecto.

class LineaTeleferico:

def \_\_init\_\_(self, color, tramo, nroCabinas):

self.color = color

self.tramo = tramo

self.nroCabinas = nroCabinas

self.nroEmpleados = 0

self.empleados = []

self.edades = []

self.sueldos = []

def agregar\_empleado(self, nombre, edad, sueldo):

self.empleados.append(nombre)

self.edades.append(edad)

self.sueldos.append(sueldo)

self.nroEmpleados += 1

def mostrar\_empleados(self):

print(f"Empleados de la línea {self.color}:")

for i in range(self.nroEmpleados):

print(f"{self.empleados[i]} - Edad: {self.edades[i]} - Sueldo: {self.sueldos[i]}")

def eliminar\_apellido(self, apellido):

i = 0

while i < len(self.empleados):

if apellido in self.empleados[i]:

del self.empleados[i]

del self.edades[i]

del self.sueldos[i]

self.nroEmpleados -= 1

else:

i += 1

def \_\_add\_\_(self, otro):

nombre = "Lucy Sosa Rios"

if nombre in self.empleados:

idx = self.empleados.index(nombre)

otro.agregar\_empleado(self.empleados[idx], self.edades[idx], self.sueldos[idx])

self.empleados.pop(idx)

self.edades.pop(idx)

self.sueldos.pop(idx)

self.nroEmpleados -= 1

print(f"Empleado '{nombre}' transferido de la línea {self.color} a la línea {otro.color}.")

else:

print(f"Empleado '{nombre}' no se encontró en la línea {self.color}.")

return otro

def mostrar\_datos(self, tipo\_filtro=None, valor=None):

print(f"color = \"{self.color}\"")

print(f"tramo = \"{self.tramo}\"")

print(f"nroCabinas = {self.nroCabinas}")

print(f"nroEmpleados = {self.nroEmpleados}")

print(f"Empleados de la línea {self.color}:")

for i in range(self.nroEmpleados):

if tipo\_filtro == "edad" and self.edades[i] >= valor:

print(f"{self.empleados[i]} - Edad: {self.edades[i]} - Sueldo: {self.sueldos[i]}")

elif tipo\_filtro == "sueldo" and self.sueldos[i] >= valor:

print(f"{self.empleados[i]} - Edad: {self.edades[i]} - Sueldo: {self.sueldos[i]}")

elif tipo\_filtro is None:

print(f"{self.empleados[i]} - Edad: {self.edades[i]} - Sueldo: {self.sueldos[i]}")

print()

# (a) Instanciar objetos

linea1 = LineaTeleferico("Rojo", "Estación Central, Cementerio, 16 de Julio", 20)

linea2 = LineaTeleferico("Azul", "Parque Mirador, Estación Busch, Del Poeta", 25)

# Agregar empleados a linea1

linea1.agregar\_empleado("Pedro Rojas Luna", 35, 2500)

linea1.agregar\_empleado("Lucy Sosa Rios", 43, 3250)

linea1.agregar\_empleado("Ana Perez Rojas", 26, 2700)

linea1.agregar\_empleado("Saul Arce Calle", 29, 2500)

# Agregar empleados a linea2

linea2.agregar\_empleado("Luis Gómez", 30, 2200)

# Mostrar empleados

print("\n--- ANTES DE TRANSFERENCIA Y ELIMINACIÓN ---")

linea1.mostrar\_datos()

linea2.mostrar\_datos()

# (b) Eliminar a los empleados con apellido X

linea1.eliminar\_apellido("Calle")

print("\n--- DESPUÉS DE ELIMINAR APELLIDO 'Calle' EN LÍNEA 1 ---")

linea1.mostrar\_datos()

# (c) Sobrecargar operador +

linea2 = linea1 + linea2

print("\n--- DESPUÉS DE TRANSFERIR 'Lucy Sosa' A LÍNEA 2 ---")

linea1.mostrar\_datos()

linea2.mostrar\_datos()

print("--- EMPLEADOS CON SUELDO >= 2500 EN LÍNEA 2 ---")

linea2.mostrar\_datos(tipo\_filtro="sueldo", valor=2500)

print("--- EMPLEADOS MAYORES DE 30 AÑOS EN LÍNEA 1 ---")

linea1.mostrar\_datos(tipo\_filtro="edad", valor=30)

SALIDA

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.  
Texto

El contenido generado por IA puede ser incorrecto.

PRIMER PARCIAL FILA B

Texto

El contenido generado por IA puede ser incorrecto.

Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Clase: Planta

* evaluar\_dispersión()

Calcula o informa cuánta capacidad tiene la planta para dispersarse en el entorno.

Ejemplo: evaluar\_dispersión() → “30 km”

* adaptarse\_condicion()

Simula cómo la planta adapta su biología a una condición ambiental (como sequía, frío).

Ejemplo: adaptarse\_condicion("sequía")

Clase: CondicionAmbiental

* cambiar\_intensidad()

Cambia la intensidad del factor ambiental (por ejemplo, sube la temperatura o aumenta la

sequía).

Ejemplo: cambiar\_intensidad("alta")

* evaluar\_impacto()

Evalúa cuánto impacto tiene la condición ambiental sobre las plantas de la zona.

Ejemplo: evaluar\_impacto() → “impacto severo”

Clase: BarrerasGeograficas

* medir\_extension()

Calcula o informa la longitud/extensión de la barrera.

Ejemplo: medir\_extension() → “500 km”

* evaluar\_restriccion()

Evalúa qué tan restrictiva es esa barrera para que las plantas puedan dispersarse.

Ejemplo: evaluar\_restriccion() → “90% de restricción”

Interfaz de usuario gráfica, Tabla

El contenido generado por IA puede ser incorrecto.

class Ministerio:

def \_\_init\_\_(self, nombre, direccion):

self.nombre = nombre

self.direccion = direccion

self.nroEmpleados = 0

self.empleados = []

self.edades = []

self.sueldos = []

def agregar\_empleado(self, nombre, edad, sueldo):

self.empleados.append(nombre)

self.edades.append(edad)

self.sueldos.append(sueldo)

self.nroEmpleados += 1

def eliminar\_por\_edad(self, edad\_x):

indices = [i for i, edad in enumerate(self.edades) if edad == edad\_x]

for i in sorted(indices, reverse=True):

self.empleados.pop(i)

self.edades.pop(i)

self.sueldos.pop(i)

self.nroEmpleados -= 1

#c) Sobrecarga del operador + para transferir empleado X al otro ministerio

def \_\_add\_\_(self, other):

nombre = "Lucy Sosa Rios"

if nombre in self.empleados:

idx = self.empleados.index(nombre)

other.agregar\_empleado(self.empleados[idx], self.edades[idx], self.sueldos[idx])

self.empleados.pop(idx)

self.edades.pop(idx)

self.sueldos.pop(idx)

self.nroEmpleados -= 1

print(f"Empleado '{nombre}' transferido de la línea {self.nombre} a la línea {other.nombre}.")

else:

print(f"Empleado '{nombre}' no se encontró en la línea {self.nombre}.")

return other

def mostrar\_todos(self):

print(f"Empleados de la línea {self.nombre}:")

for i in range(self.nroEmpleados):

print(f"{self.empleados[i]} - Edad: {self.edades[i]} - Sueldo: {self.sueldos[i]}")

print()

def mostrar\_menor\_edad(self):

if not self.edades:

return

menor\_edad = min(self.edades)

print(f"Empleados de la línea {self.nombre} con la menor edad:")

for i in range(self.nroEmpleados):

if self.edades[i] == menor\_edad:

print(f"{self.empleados[i]} - Edad: {self.edades[i]} - Sueldo: {self.sueldos[i]}")

print()

def mostrar\_menor\_sueldo(self):

if not self.sueldos:

return

menor\_sueldo = min(self.sueldos)

print(f"Empleados de la línea {self.nombre} con el menor sueldo:")

for i in range(self.nroEmpleados):

if self.sueldos[i] == menor\_sueldo:

print(f"{self.empleados[i]} - Edad: {self.edades[i]} - Sueldo: {self.sueldos[i]}")

print()

#a) Instanciar 2 objetos Ministerio de diferente forma

linea\_rojo = Ministerio("Rojo", "Estación Central")

linea\_azul = Ministerio("Azul", "Estación 16 de Julio")

#Agregar empleados al ministerio Rojo

linea\_rojo.agregar\_empleado("Pedro Rojas Luna", 35, 2500)

linea\_rojo.agregar\_empleado("Lucy Sosa Rios", 43, 3250)

linea\_rojo.agregar\_empleado("Ana Perez Rojas", 26, 2700)

linea\_rojo.agregar\_empleado("Saul Arce Calle", 29, 2500)

# Agregar empleados al ministerio Azul

linea\_azul.agregar\_empleado("Luis Gómez", 30, 2200)

# Mostrar estado inicial

print("--- ANTES DE TRANSFERIR ---")

linea\_rojo.mostrar\_todos()

linea\_azul.mostrar\_todos()

#c) Transferir 'Lucy Sosa Rios' de Rojo a Azul

linea\_rojo + linea\_azul

print("--- DESPUÉS DE TRANSFERIR 'Lucy Sosa Rios' A LÍNEA 2 ---")

linea\_rojo.mostrar\_todos()

linea\_azul.mostrar\_todos()

#b) Eliminar empleados con edad X (ejemplo: eliminar edad 29 en Rojo)

linea\_rojo.eliminar\_por\_edad(29)

print("--- DESPUÉS DE ELIMINAR EMPLEADOS DE 29 AÑOS EN LÍNEA ROJO ---")

linea\_rojo.mostrar\_todos()

#d1) Mostrar empleados con la menor edad

print("--- EMPLEADOS CON LA MENOR EDAD EN LÍNEA ROJO ---")

linea\_rojo.mostrar\_menor\_edad()

print("--- EMPLEADOS CON LA MENOR EDAD EN LÍNEA AZUL ---")

linea\_azul.mostrar\_menor\_edad()

#d2) Mostrar empleados con el menor sueldo

print("--- EMPLEADOS CON EL MENOR SUELDO EN LÍNEA ROJO ---")

linea\_rojo.mostrar\_menor\_sueldo()

print("--- EMPLEADOS CON EL MENOR SUELDO EN LÍNEA AZUL ---")

linea\_azul.mostrar\_menor\_sueldo()

SALIDA

Texto

El contenido generado por IA puede ser incorrecto.  
Texto

El contenido generado por IA puede ser incorrecto.