

PROGRAMACIÓN DE SISTEMAS EMBUDIDOS

PRACTICA HITO 2
ESTUDIANTE: JOSUE O. PORCEL M.

CONTENIDO DE LA PRÁCTICA

Esta tarea se divide en **dos** partes:

- La primera parte corresponde a la *parte TEÓRICA* necesaria, en donde se encuentra un conglomerado de preguntas relacionadas a SISTEMAS EMBEBIDOS Y PYTHON.
- La segunda parte corresponde a la *parte PRÁCTICA* necesaria en donde deberá resolver ejercicios de programación usando el lenguaje de Python.

TABLA DE CONTENIDO

O1

MANEJO DE
CONCEPTOS

O2

PARTE
PRÁCTICA



01

MANEJO DE CONCEPTOS

¿QUÉ ES UN SISTEMA EMBEBIDO?

Un sistema embebido o empotrado a un sistema electrónico de cómputo diseñado para realizar o ejecutar una o varias tareas o funciones específicas

¿MENCIONE 5 SISTEMAS EMBEBIDOS?

- Un Microondas
- Un Lavadora
- Equipo de Música
- Sistema GPS
- Sistema de Control de Acceso



¿MENCIONA LAS DIFERENCIAS O SIMILITUDES ENTRE UN SISTEMA OPERATIVO, UN SISTEMA MÓVIL Y UN SISTEMA EMBEBIDO?

Los tres tipos de sistemas en común está el administrar o controlar según las necesidades (SO programas de un computadora, SO Móvil gestiona el hardware haciendo posible un manejo de los smartphone, SE para dispositivos electrónicos y/o mecánicos) porque el nivel de administración de cada uno es muy diferente, un SO puede con equipos de gran tamaño, los SO móviles con equipos de menor tamaño además de jugar con el Hardware y los SE equipos de aun menor capacidad.



¿A QUÉ SE REFERIRÁN LOS TÉRMINOS MCU Y MPU? EXPLIQUE CADA UNA DE ELLAS

- MPU hace referencia al Microprocesador, que se encuentra integrado en la placa base y que se encarga de ejecutar las instrucciones que ordena el usuario.
- MCU hace referencia al Microcontrolador, el cual es un circuito integrador programable, en términos simples hablamos de una pequeña computadora



¿CUÁLES SON LOS PILARES DE POO?

Encapsulación, Abstracción, Herencia, Polimorfismo

¿MENCIONE LOS COMPONENTES EN LO QUE SE BASA POO?

- Clase.- Es donde estarán guardados los objetos del programa.
- Objeto.- Es un ente usado el cual nos permite separar los diferentes componentes del programa de tal manera ayuda a identificar y simplificar su elaboración.
- Instancia.- No es mas que hacer una referencia a una clase para que puedan interactuar entre ellas



¿CUÁLES SON LOS PILARES DE POO?

Encapsulación, Abstracción, Herencia, Polimorfismo

¿MENCIONE LOS COMPONENTES EN LO QUE SE BASA POO?

- Clase.- Es donde estarán guardados los objetos del programa.
- Objeto.- Es un ente usado el cual nos permite separar los diferentes componentes del programa de tal manera ayuda a identificar y simplificar su elaboración.
- Instancia.- No es mas que hacer una referencia a una clase para que puedan interactuar entre ellas



DEFINA LOS SIGUIENTES CONCEPTOS:

- **MULTIPLATAFORMA.-** Un software que puede existir en varios sistemas operativos o dispositivos móviles.
- **MULTIPARADIGMA.-** Que soporta o posee características que estén relacionados hacia la programación orientada a objetos, que sean imperativos, declarativo, lógico y funcional, que puede ser escrito de múltiples maneras.
- **MULTIPROPOSITO.-** Que es aplicable en varios campos de la industria, sea la IA, hacking entre otros.
- **LENGUAJE INTERPRETADO.-** Que puede ser interpretado o compilado con un software el cual le ha de comunicar las instrucciones a la computadora y esta la logre entender.



DEFINA A QUE SE REFIERE CUANDO SE HABLA DE ENCAPSULACIÓN Y MUESTRE UN EJEMPLO (CÓDIGO EN PYTHON)

Ocultamos información, haciendo visible solo aquello que es indispensable para el ojo del usuario y dejando en oculto todo lo demás.

Ejemplo Calculadora con Signos A B





```
1 n1 = input("Introduzca un Valor entre A, B")
2 rs = input("Introduzca la funcion suma (+)")
3 n2 = input("Introduzca un Valor entre A, B")
4
5 if n1 == "A":
6     a1 = 2
7 elif n1 == "B":
8     a1 = 6
9 if n2 == "A":
10    b2 = 2
11 elif n2 == "B":
12    b2 = 6
13
14 if rs == "+":
15     print(n1, rs, n2, '=')
16     if a1 == b2:
17         if a1 == 2:
18             print('Resp A')
19         else:
20             print('Resp B')
21     else:
22         sum = a1 + b2
23         if sum == 8:
24             print('Resp F')
```

RESULTADOS

```
Interactive-1 X ...
X Clear All Restart Interrupt Variables Save ... Python 3.10.2 64-bit

✓ n1 = input("Introduzca un Valor entre A, B") ...
... A + A =
Resp A

✓ n1 = input("Introduzca un Valor entre A, B") ...
... B + B =
Resp B

✓ n1 = input("Introduzca un Valor entre A, B") ...
... A + B =
Resp F
```

DEFINA A QUE SE REFIERE CUANDO SE HABLA DE HERENCIA Y MUESTRE UN EJEMPLO (CÓDIGO EN PYTHON).

Es una forma de reutilizar el código, ya que existen clases que comparten características similares de tal modo se crea una clase madre o superclase la cual dará a las clases que son iguales atributos que ella tiene.

Ejemplo un Gerente y un Empleado



```

1 class Person:
2     fullname = None
3     lastname = None
4     age = None
5     email = None
6
7     def __init__(self, fullname, lastname, age, email):
8         self.fullname = fullname
9         self.lastname = lastname
10        self.age = age
11        self.email = email
12
13    def __str__(self):
14        return f'Nombre: {self.fullname}\nApellido: {self.lastname}\nEdad: {self.age}\nEmail: {self.email}\n'
15
16 class Gerente:
17     charge = None
18     numb_emp_in_char = None
19
20     def __init__(self, fullname, lastname, age, email, charge, numb_emp_in_char):
21         Person.__init__(self, fullname, lastname, age, email)
22         self.charge = charge
23         self.numb_emp_in_char = numb_emp_in_char
24
25     def __str__(self):
26         return Person.__str__(self) + f'Cargo: {self.charge}\nCantidad de Personas a su Cargo: {self.numb_emp_in_char}'
27
28 class Empleado:
29     code = None
30     desinated_area = None
31
32     def __init__(self, fullname, lastname, age, email, code, desinated_area):
33         Person.__init__(self, fullname, lastname, age, email)
34         self.code = code
35         self.desinated_area = desinated_area
36
37     def __str__(self):
38         return Person.__str__(self) + f'Codigo de Empleado: {self.code}\nArea designada: {self.desinated_area}'
39
40 print('Gerente')
41 ger1 = Gerente('Tomas', 'Diaz', 32, 'asd@gmail.com', 'Doctor', 15)
42 print(ger1)
43 print(' ')
44 print("Empleado")
45 emp1 = Empleado('Martha', 'Kane', 28, 'dsa@gmail.com', 'ENF-001', 'Enfermeria')
46 print(emp1)

```

RESULTADOS

CodeSnap Interactive-1 X

Clear All Restart Interrupt Variables Save Export Expand Python 3.10.2 64-bit

```

✓ class Person: ...
...
Gerente
Nombre: Tomas
Apellido: Diaz
Edad: 32
Email: asd@gmail.com
Cargo: Doctor
Cantidad de Personas a su Cargo: 15

Empleado
Nombre: Martha
Apellido: Kane
Edad: 28
Email: dsa@gmail.com
Codigo de Empleado: ENF-001
Area designada: Enfermeria

```



O2

PARTE
PRÁCTICA

LLEVAR EL SIGUIENTE CÓDIGO JAVA A PYTHON



```
1  class Main{
2      public static void main(String[] args){
3
4          System.out.println("Enter two numbers");
5          int first = 10;
6          int second = 20;
7
8          System.out.println(first + " + " + second);
9
10         //add two numbers
11         int sum = first + second;
12         System.out.println("The sum is: "+sum);
13     }
14 }
```


LLEVAR EL SIGUIENTE CÓDIGO JAVA A PYTHON



```
1  print('Enter two Numbers')
2  a1, b1 = 10, 20
3
4  print(a1, '+', b1)
5
6  #add two numbers
7  sum = a1 + b1
8  print('The sum is: ',sum)
```

RESULTADOS

The screenshot shows a Python 3.10.2 64-bit interactive shell window. The code from the previous block has been executed. The output is as follows:

```
✓ print('Enter two Numbers') ...
... Enter two Numbers
    10 + 20
    The sum is:  30
```

CREAR EL CÓDIGO JAVA Y PYTHON PARA EL SIGUIENTE ANÁLISIS.



Propiedad

name
email
gender
nationality

Comportamiento

Write book
Write a movie
Change nationality
Change email

CREAR EL CÓDIGO JAVA Y PYTHON PARA EL SIGUIENTE ANÁLISIS.

```
1 public class Pruebas_Doce {
2     String name;
3     String email;
4     String gender;
5     String nationality;
6
7     public static void main(String[] args) {
8         Pruebas_Doce doce = new Pruebas_Doce();
9         doce.name = "William";
10        doce.email = "asdk@gmail.com";
11        doce.gender = "Masculine";
12        doce.nationality = "Bolivian";
13
14        System.out.print("Nombre: " +doce.name+
15        " Email: " + doce.email+" Genero "+doce.gender+
16        " Nacionalidad: "+doce.nationality);
17
18    }
19 }
```

RESULTADOS

```
[Running] cd "d:\PSE\Proyectos-de-PSE\Hito2\Practica Hito 2\" && javac Pruebas
Nombre: William Email: asdk@gmail.com Genero Masculine Nacionalidad: Bolivian
[Done] exited with code=0 in 0.775 seconds
```

CREAR EL CÓDIGO JAVA Y PYTHON PARA EL SIGUIENTE ANÁLISIS.

```
1 class Person:
2     name = None
3     email = None
4     gender = None
5     nationality = None
6
7     def __init__(self, name, email, gender, nationality):
8         self.name = name
9         self.email = email
10        self.gender = gender
11        self.nationality = nationality
12
13    def __str__(self):
14        return f'Name: {self.name} \nEmail: {self.email} \nGender: {self.gender} \nNationality: {self.nationality} \n'
15
16    def Write_book(self):
17        pass
18    def Write_a_movies(self):
19        pass
20
21    def change_nationality(self, newnationality):
22        self.nationality = newnationality
23
24    def change_email(self, newemail):
25        self.email = newemail
26
27    per1 = Person('William', 'asdk@gmail.com', 'Masculine', "Bolivian")
28    print(per1)
29
30
31    per1.change_nationality('Mexican')
32    per1.change_email('dfgk@gmail.com')
33    print(per1)
```

RESULTADOS

✓ `class Person: ...`

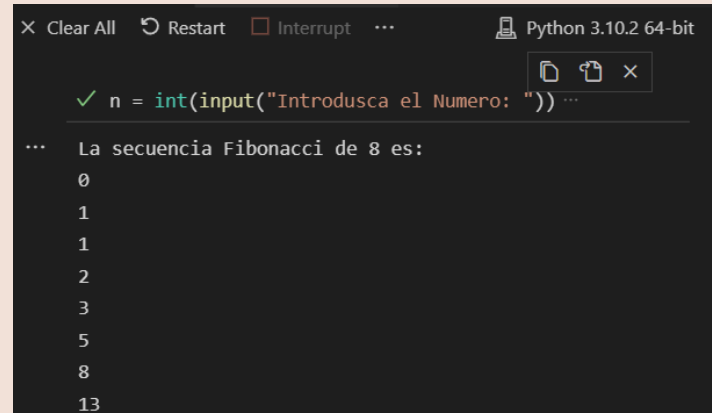
Name: William
Email: asdk@gmail.com
Gender: Masculine
Nationality: Bolivian

Name: William
Email: dfgk@gmail.com
Gender: Masculine
Nationality: Mexican

CREAR UN PROGRAMA PYTHON QUE GENERE LOS PRIMEROS N NÚMEROS DE LA SERIE FIBONACCI.

```
1  n = int(input("Introduzca el Numero: "))
2  n1 = 0
3  n2 = 1
4  count = 0
5  if n <= 0:
6      print("Introduce un numero mayor a 0 ")
7  elif n <= 1:
8      print("La secuencia Fibonacci de", n , "es:", n, ":")
9  else:
10     print("La secuencia Fibonacci de", n , "es:")
11     while count < n:
12         print(n1)
13         sum = n1 + n2
14         n1 = n2
15         n2 = sum
16         count += 1
```

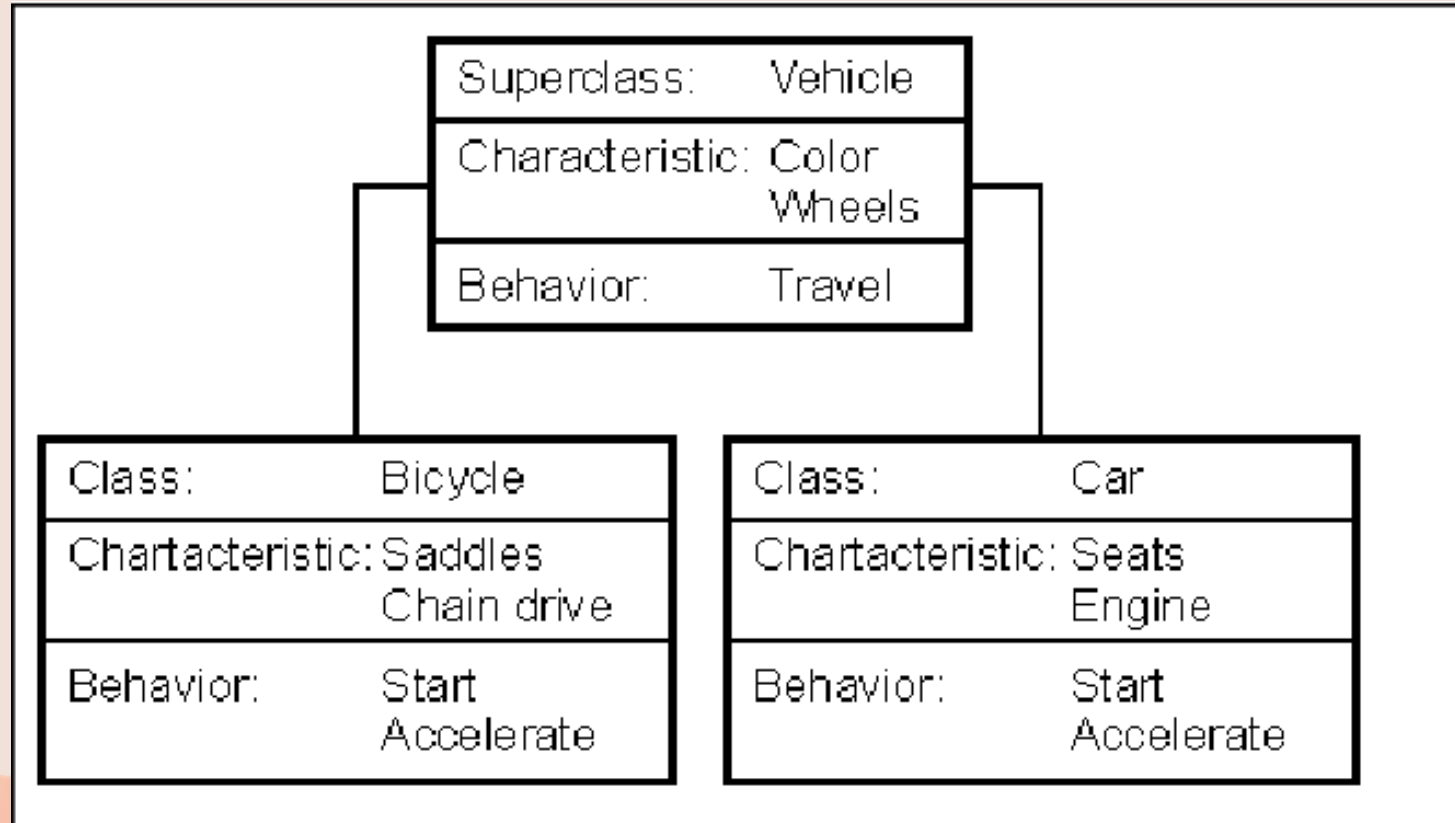
RESULTADOS



The screenshot shows a Python IDE window titled "Python 3.10.2 64-bit". The code being executed is the same as in the first block. The output shows the sequence of Fibonacci numbers for n=8: 0, 1, 1, 2, 3, 5, 8, 13.

```
✓ n = int(input("Introduzca el Numero: ")) ...
... La secuencia Fibonacci de 8 es:
0
1
1
2
3
5
8
13
```

CREAR LAS CLASES NECESARIAS PARA RESOLVER EL SIGUIENTE PLANTEAMIENTO



CREAR LAS CLASES NECESARIAS PARA RESOLVER EL SIGUIENTE PLANTEAMIENTO

```
1 class Vehicle:
2     color = None
3     wheels = None
4
5     def __init__(self, color, wheels):
6         self.color = color
7         self.wheels = wheels
8
9     def __str__(self):
10         return f'Color: {self.color}\n Wheels: {self.wheels}\n'
11
12     def travel(self):
13         pass
```

CREAR LAS CLASES NECESARIAS PARA RESOLVER EL SIGUIENTE PLANTEAMIENTO



```
1  class Cars(Vehicle):
2      sets = None
3      engine = None
4
5      def __init__(self, color, wheels, sets, engine):
6          Vehicle.__init__(self, color, wheels)
7          self.sets = sets
8          self.engine = engine
9
10     def __str__(self):
11         return Vehicle.__str__(self) + f'Sets: {self.sets}\n Engine: {self.engine}\n'
12
13     def start(self):
14         pass
15     def accelerate(self):
16         pass
```


CREAR LAS CLASES NECESARIAS PARA RESOLVER EL SIGUIENTE PLANTEAMIENTO

```
1 class Bicycles(Vehicle):
2     saddles = None
3     chain_drive = None
4
5     def __init__(self,color, wheels, saddles, chain_drive):
6         Vehicle.__init__(self, color, wheels)
7         self.saddles = saddles
8         self.chain_drive = chain_drive
9
10    def __str__(self):
11        return Vehicle.__str__(self) + f'Saddles: {self.saddles}\nChain Drive {self.chain_drive}'
12
13    def start(self):
14        pass
15    def accelerate(self):
16        pass
```

CREAR LAS CLASES NECESARIAS PARA RESOLVER EL SIGUIENTE PLANTEAMIENTO

MAIN

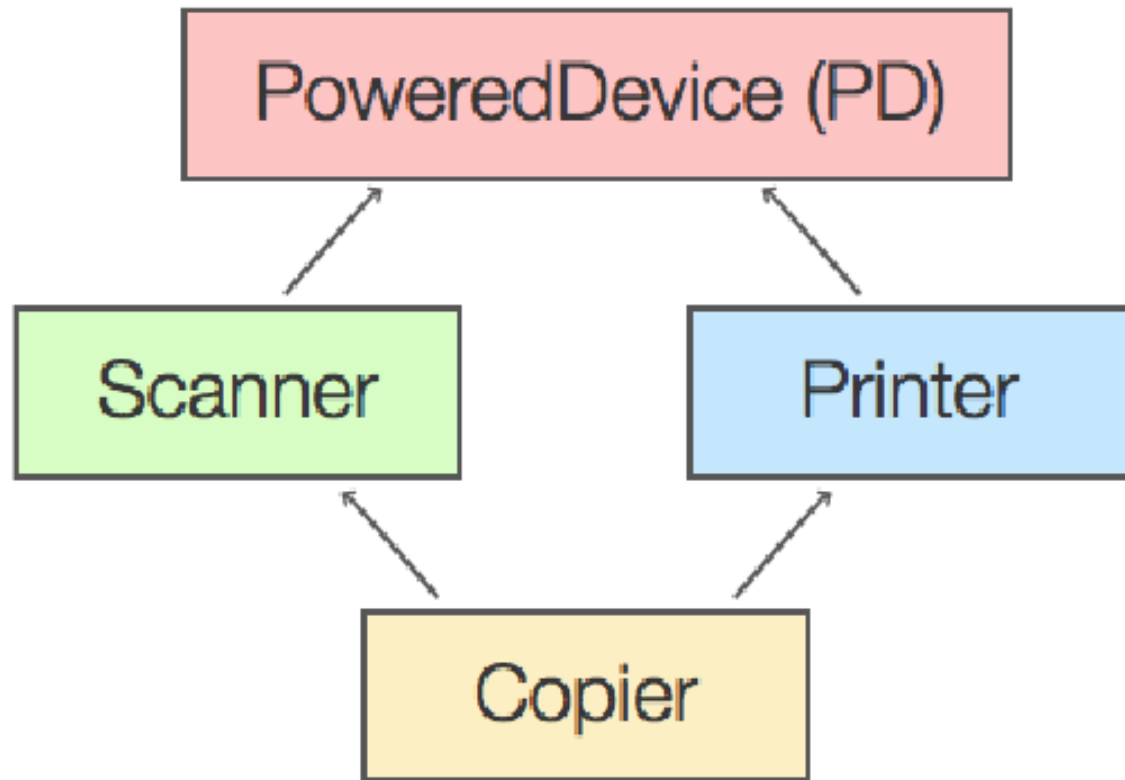
```
1 print('CAR')
2 car1 = Cars('Red', 'Four', 'Five', 'Motor Wankel')
3 print(car1)
4
5 print('BICYCLE')
6 bic1 = Bicycles('Black', 'Two', 'One', 'Seven')
7 print(bic1)
```

RESULTADOS

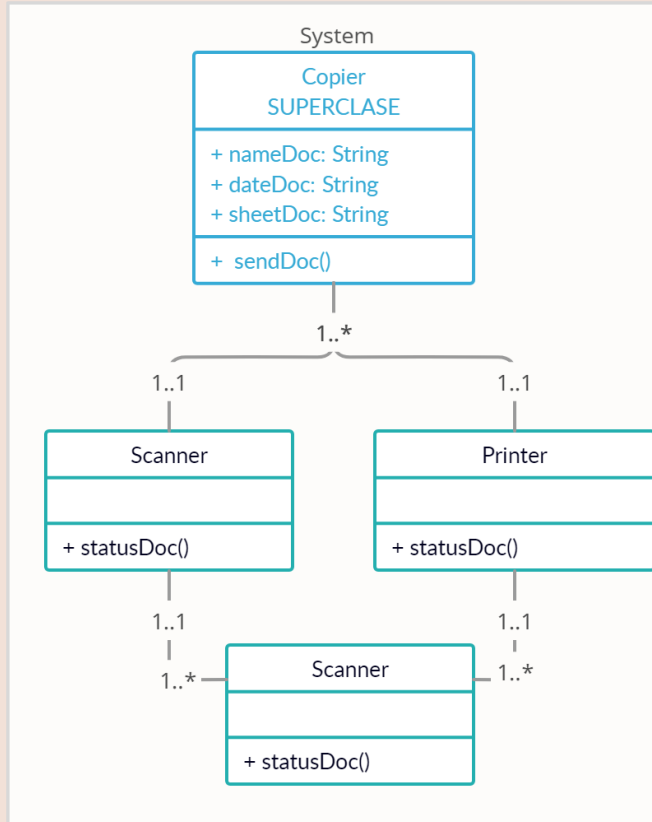
```
✓ class Vehicle: ...
... CAR
  Color: Red
  Wheels: Four
  Sets: Five
  Engine: Motor Wankel

  BICYCLE
  Color: Black
  Wheels: Two
  Saddles: One
  Chain Drive Seven
```

REALIZAR UN ANÁLISIS PARA EL SIGUIENTE ESCENARIO



REALIZAR UN ANÁLISIS PARA EL SIGUIENTE ESCENARIO



REALIZAR UN ANÁLISIS PARA EL SIGUIENTE ESCENARIO

```
1 class Copier:
2     nameDoc = None
3     dateDoc = None
4     sheetsDoc = None
5
6     def __init__(self, nameDoc, dateDoc, sheetsDoc):
7         self.nameDoc = nameDoc
8         self.dateDoc = dateDoc
9         self.sheetsDoc = sheetsDoc
10
11     def __str__(self):
12         return f'Nombre del Documento: {self.nameDoc}\nFecha del Documento: {self.dateDoc}\nCantidad de Hojas: {self.sheetsDoc}\n'
13
14     def senddoc(self):
15         print("Documentos Enviados")
```

REALIZAR UN ANÁLISIS PARA EL SIGUIENTE ESCENARIO

```
1 class Scanner(Copier):
2     def __init__(self, nameDoc, dateDoc, sheetsDoc):
3         Copier.__init__(self, nameDoc, dateDoc, sheetsDoc)
4
5     def __str__(self):
6         return Copier.__str__(self)
7
8     def statusDoc(self):
9         print("Documento Escaneado")
10
11 class Printer:
12     def __init__(self, nameDoc, dateDoc, sheetsDoc):
13         Copier.__init__(self, nameDoc, dateDoc, sheetsDoc)
14
15     def __str__(self):
16         return Copier.__str__(self)
17
18     def statusDoc(self):
19         print("Documento Imprimido")
```

REALIZAR UN ANÁLISIS PARA EL SIGUIENTE ESCENARIO

RESULTADOS

```
1 class PD(Scanner, Copier):
2     def __init__(self):
3         print('Dispositivos Alimentados')
4
5 scanner1 = Scanner('Proyectos', '02/03/2022', 78)
6 print(scanner1)
7 printer1 = Printer('Hojas de Vida', '02/03/2022', 3)
8 print(printer1)
9 pd1 = PD()
10 pd1.statusDoc()
```

```
✓ class Copier: ...
```

```
... Nombre del Documento: Proyectos
    Fecha del Documento: 02/03/2022
    Cantidad de Hojas: 78
```

```
Nombre del Documento: Hojas de Vida
Fecha del Documento: 02/03/2022
Cantidad de Hojas: 3
```

```
Dispositivos Alimentados
Documento Escaneado
```

EJERCICIO DE PLANTEAMIENTO.

```
1 class Person:
2     name = None
3     age = None
4     email = None
5     direction = None
6
7     def __init__(self, name, age, email, direction):
8         self.name = name
9         self.age = age
10        self.email = email
11        self.direction = direction
12
13    def __str__(self):
14        return f'Nombre: {self.name}\nEdad: {self.age}\nCorreo: {self.email}\n'
15
16    def observations (self, observation):
17        self.observations = observation
18        print(observations)
```


EJERCICIO DE PLANTEAMIENTO.

```
1  class Student(Person):
2      codeStudent = None
3
4      def __init__(self, codeStudent, name, age, email, direction):
5          Person.__init__(self, name, age, email, direction)
6          self.codeStudent = codeStudent
7
8      def __str__(self):
9          return f'Codigo: {self.codeStudent}\n' + Person.__str__(self)
10
11
12     def qualification (self, note):
13         self.qualification = note
14         print(note)
```

EJERCICIO DE PLANTEAMIENTO.



```
1  class Manager(Person):
2      codeManager = None
3      department = None
4
5      def __init__(self, codeManager, department, name, age, email, direction):
6          Person.__init__(self, name, age, email, direction)
7          self.codeManager = codeManager
8          self.department = department
9
10     def __str__(self):
11         return f'Departamento: {self.department}\n' + Person.__str__(self)
```

EJERCICIO DE PLANTEAMIENTO.



```
1 class dateXtra:
2     info = None
3     semester = None
4
5     def __init__(self, info, semester):
6         self.info = info
7         self.semester = semester
8
9     def __str__(self):
10         return f'Informacion: {self.info}\nSemestre: {self.semester}'
```

EJERCICIO DE PLANTEAMIENTO.



```
1  class SM(Student, dateXtra):
2      key = None
3
4      def __init__(self, key, codeStudent, name, age, email, direction, info, semester):
5          Student.__init__(self, codeStudent, name, age, email, direction)
6          dateXtra.__init__(self, info, semester)
7          self.key = key
8
9      def __str__(self):
10         return f'SM: {self.key}\n' + Student.__str__(self) + dateXtra.__str__(self)
11
12     def observations(self, observation):
13         self.observations = observation
14         print(observation)
```

EJERCICIO DE PLANTEAMIENTO.



```
1 mn1 = Manager('Cod-001', 'Turismo', 'William', 32, 'willi@gmail.com', 'Av.Civica')
2 print(mn1)
3
4 st1 = Student('SIS13298376', 'Freddy', 25, 'fred@gmail.com', 'Av.Argelia')
5 print(st1)
6
7 sm1 = SM('SIS-TRM', 'SIS13298376', 'Vania', 18, 'vangrihs@gmail.com', 'Av. Policia', 'Sin Comentarios', 6)
8 print(sm1)
```

EJERCICIO DE PLANTEAMIENTO.

✓ `class Person: ...`

Departamento: Turismo

Nombre: William

Edad: 32

Correo: willi@gmail.com

Codigo: SIS13298376

Nombre: Freddy

Edad: 25

Correo: fred@gmail.com

SM: SIS-TRM

Codigo: SIS13298376

Nombre: Vania

Edad: 18

Correo: vangrihs@gmail.com

Informacion: Sin Comentarios

Semestre: 6