



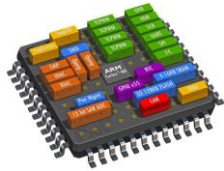
PROGRAMACIÓN EN SISTEMAS EMBEBIDOS

ANTONIO MANUEL GONZALES TICONA

Preguntas Teóricas 1-5.

Pregunta 1.

¿Qué es un sistema embebido?



Pregunta 2.

Mencione 5 ejemplos de sistemas embebidos.



Pregunta 3.

Menciona las diferencias o similitudes entre un sistema operativo, un sistema móvil y un sistema embebido.

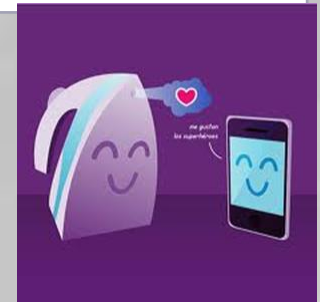
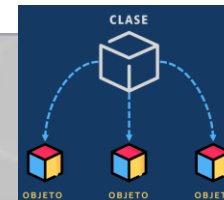
Pregunta 4.

¿A qué se refieren los términos MCU y MPU?



Pregunta 5.

¿Cuáles son los pilares de POO?

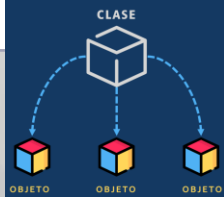


Preguntas Teóricas 6-10.

Orientado a Objetos	Imperativo	Lógico	Funcional	Orientado a Aspectos
Ada	•			
C++	•			
Ruby	•		•	
Python	•		•	•
CIAO	•	•		
Leda	•	•	•	

Pregunta 6.

Mencione los componentes en los que se basa POO. Y explicar cada una de ellas.

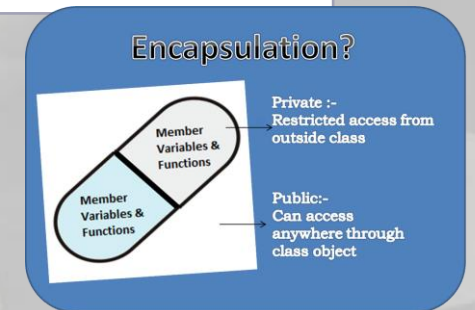


Pregunta 7.

Defina: Multiplataforma, Multiparadigma, Multipropósito, Lenguaje interpretado.

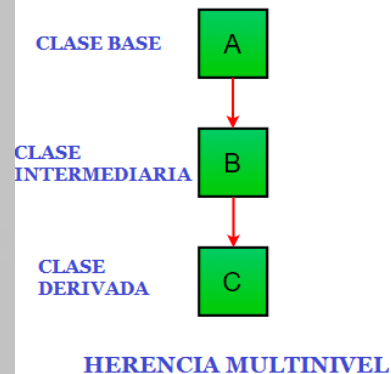
Pregunta 8.

Defina a que se refiere cuando se habla de encapsulación y muestre un ejemplo.



Pregunta 9.

Defina a que se refiere cuando se habla de herencia y muestre un ejemplo.



Pregunta 10.

Defina los siguientes: Clase, Objeto, Instancia.

Parte Práctica.

11. Convertir el siguiente código JAVA a Python.

```
class Main {  
  
    public static void main(String[] args) {  
  
        System.out.println("Enter two numbers");  
        int first = 10;  
        int second = 20;  
  
        System.out.println(first + " " + second);  
  
        // add two numbers  
        int sum = first + second;  
        System.out.println("The sum is: " + sum);  
    }  
}
```

Convertido a PYTHON

```
n1 = int(input("Ingrese el primer número:"))  
n2 = int(input("Ingrese el segundo número: "))  
sum = n1 + n2  
print("La suma es: ", sum)
```

```
Ingrese el primer número:5  
Ingrese el segundo número: 2  
La suma es: 7
```

```

public class Persona
{
    private String nombre;
    private String email;
    private String genero;
    private String nationality;
    // constructor
    public Persona(String name ,String email,String genero,String
nacionalidad)
    {
        this.nombre = name;
        this.email = email;
        this.genero = genero;
        this.nationality = nacionalidad;
    }

    //Métodos
    public void escribeLibro()
    {
        System.out.println("Escribió un libro");
    }
    public void escribePelícula()
    {
        System.out.println("Escribió una película");
    }

    //Método para establecer nacionalidad
    public void setNationality(String nuevaNatio)
    {
        nationality = nuevaNatio;
    }

    //Método para establecer nuevo email
    public void setEmail(String nuevoEmail)
    {
        email = nuevoEmail;
    }
}

```



Parte Práctica.

- 12. 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

Convertido a PYTHON

```
class Persona:
    __name = None
    __email = None
    __gender = None
    __nationality = None

    def __init__(self, name, email, gender, nationality):
        self.__name = name
        self.__email = email
        self.__gender = gender
        self.__nationality = nationality

    #sobreescribe al metodo self y lo imprime
    def __str__(self): ## \n salto de linea --> alt+92 = \
        return f'Nombre: {self.__name} \nEmail: {self.__email} \nGénero: {self.__gender} \nNacionalidad: {self.__nationality}'

    #encapsulacion
    ## getters
    def get_nombre(self):
        return self.__name

    ## setters
    def set_nombre(self, nuevo_nombre):
        self.__name = nuevo_nombre

    ##Métodos:
    def write_book(self, name ):
        print('Ha escrito un libro')

    def write_movie(self, name ):
        print('Ha escrito una película')

    def change_nationality(self, new_nationality ):
        self.__nationality = new_nationality
        print('Se ha cambiado de nacionalidad')

    def change_email(self, new_email ):
        self.__email = new_email
        print('Se ha cambiado el Email.')

persona1 = Persona('Antonio', 'AntGonz@gmail.com', 'Masculino', 'Boliviano')
print(persona1)
persona1.change_nationality('Cubano')
print(persona1)
persona1.change_email('EmailNuevo@gmail.com')
print(persona1)
```

```
Nombre: Antonio
Email: AntGonz@gmail.com
Género: Masculino
Nacionalidad: Boliviano
Se ha cambiado de nacionalidad
```

```
Nombre: Antonio
Email: AntGonz@gmail.com
Género: Masculino
Nacionalidad: Cubano
Se ha cambiado el Email.
Nombre: Antonio
Email: EmailNuevo@gmail.com
Género: Masculino
Nacionalidad: Cubano
```

Parte Práctica.

- 12. 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

Fibonacci

```
class Series_and_strings:
```

```
def __init__(self):
```

```
    n = None
```

```
def fibonacci(self):
```

```
    n = int(input("Ingresa N: "))
```

```
    n1 = 0
```

```
    n2 = 1
```

```
    aux = 0
```

```
    print("Secuencia:")
```

```
    while aux < n:
```

```
        print(n1)
```

```
        r = n1 + n2
```

```
        n1 = n2
```

```
        n2 = r
```

```
        aux += 1
```

```
Ingresa N: 6
```

```
Secuencia:
```

```
0
```

```
1
```

```
1
```

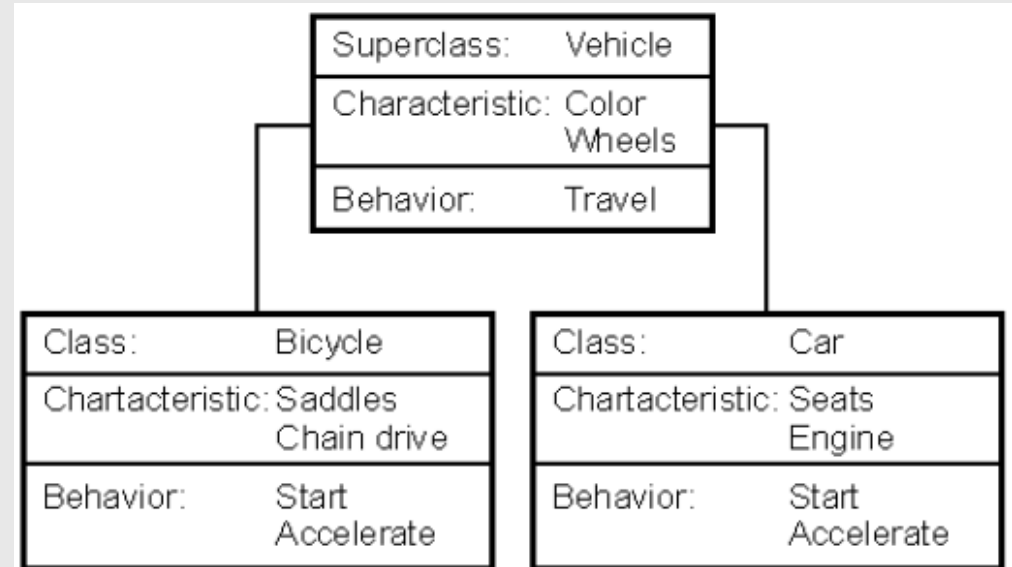
```
2
```

```
3
```

```
5
```

Parte práctica.

- 13. crear un Programa Python que genere los primeros N números de la serie Fibonacci.
- 14. Crear las clases necesarias para resolver el siguiente planteamiento.



14. Vehiculo

```
class Vehicle:
    Color = None
    Wheels = None

    def __init__(self, color, wheels):
        self.Color = color
        self.Wheels = wheels
```

```
    def __str__(self): ## \n salto de linea --> alt+92 = \
        return f'Color: {self.Color} \nLlantas: {self.Wheels} \n'
```

```
    def travel(self):
        print("Ha viajado el auto")
```

```
Vehicle1 = Vehicle('Blanco','5 Repuesto')
print(Vehicle1)
Vehicle1.travel()
```

```
class Bicycle(Vehicle):
    saddles = None
    chain_drive = None

    def __init__(self, color,wheels,saddles,chain):
        Vehicle.__init__(self, color, wheels)
        self.saddles = saddles
        self.chain_drive = chain
```

```
    def __str__(self): ## \n salto de linea --> alt+92 = \
        return f'Saddles: {self.saddles} \nChainDrive: {self.chain_drive} \n'
```

```
    def Start(self):
        print("Ha iniciado marcha")
```

```
    def Accerelate(self):
        print("Comenzó a acelerar")
```

```
Bicycle1 = Bicycle('Rojo','2','BMX','BMXPRO')
print(Bicycle1)
Bicycle1.Start()
Bicycle1.Accerelate()
```

```
class Car(Vehicle):
    seats = None
    engine = None

    def __init__(self, color, wheels, seats, engine):
        Vehicle.__init__(self, color, wheels)
        self.seats = seats
        self.engine = engine

    def __str__(self): ## \n salto de linea --> alt+92 = \
        return f'Seats: {self.seats} \nEngine: {self.engine} \n'
```

```
    def Start(self):
        print("Ha iniciado marcha (auto)")

    def Accerelate(self):
        print("Comenzó a acelerar (auto)")
```

```
Color: Blanco
Llantas: 5 Repuesto
```

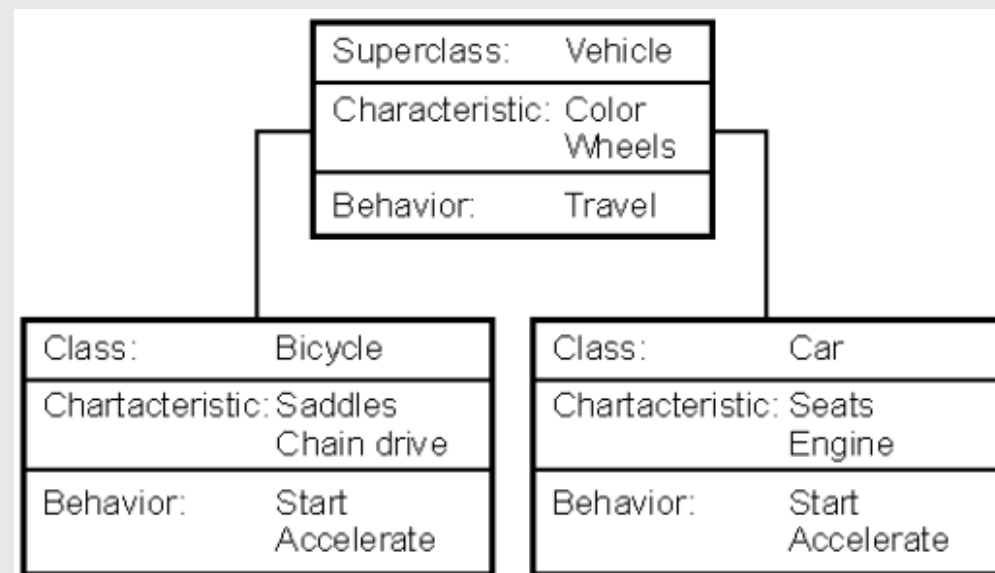
```
Ha viajado el auto
Saddles: BMX
ChainDrive: BMXPRO
```

```
Ha iniciado marcha
Comenzó a acelerar
Seats: SPARCO
Engine: Twin-turbo V6
```

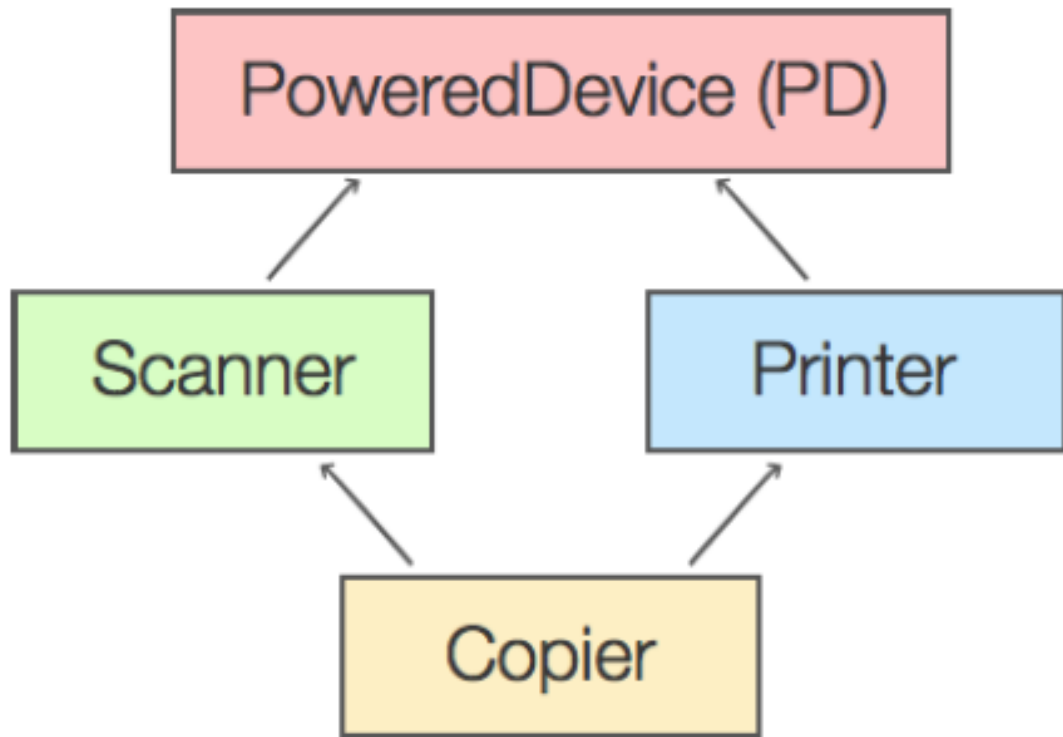
```
Ha iniciado marcha (auto)
Comenzó a acelerar (auto)
```

Parte práctica.

- 13. crear un Programa Python que genere los primeros N números de la serie Fibonacci.
- 14. Crear las clases necesarias para resolver el siguiente planteamiento.



15. Realizar un análisis para el siguiente escenario.



IDENTIFICAR
ATRIBUTOS



IDENTIFICAR MÉTODOS.



DEMÁS PARTES DE TU
ANÁLISIS

```
class PoweredDevice:
    modelo = None
    marca = None
    wifi = None
    lan = None
    usb = None

    def __init__(self, modelo, marca, wf, lan, usb):
        self.modelo = modelo
        self.marca = marca
        self.wifi = wf
        self.lan = lan
        self.usb = usb

    def __str__(self): ## \n salto de linea --> alt+92 = \
        return f'Modelo: {self.modelo} \nMarca: {self.marca} \nWifi: {self.wifi} \nLan: {self.lan} \nUsb: {self.usb} \n'

    def encender(self):
        print("Se ha encendido")

    def apagar(self):
        print("Se ha apagado")
```

```
PoweredDevice1 = PoweredDevice('L3150','EPSON','EpsonSeries','UTP','2')
print(PoweredDevice1)
PoweredDevice1.encender()
```

```
class Scanner(PoweredDevice):
    tipoVidrio = None
    Botones = None

    def __init__(self, modelo, marca, wf, lan, usb, tipo, btn):
        PoweredDevice.__init__(self, modelo, marca, wf, lan, usb)
        self.tipoVidrio = tipo
        self.Botones = btn

    def __str__(self): ## \n salto de linea --> alt+92 = \
        return f'TipoVidrio: {self.tipoVidrio} \nBotones: {self.Botones} \n'
```

```
    def Scanear(self):
        print("Ha iniciado el scaneo")

Scanner1 = Scanner('Scan1','SacnnerpRo','NO','SI','SI','VidrioTipo1','Cuatro')
print(Scanner1)
Scanner1.Scanear()
```

Modelo: L3150
Marca: EPSON
Wifi: EpsonSeries
Lan: UTP
Usb: 2

Se ha encendido

TipoVidrio: VidrioTipo1
Botones: Cuatro

Ha iniciado el scaneo
TipoHojas: Carta-Oficio
Funciones: B/N-Color

Ha iniciado las impresiones
Cantidad Maxima: Max:1000Hojas

Ha iniciado la copia BN
Ha iniciado la copia color
Se ha apagado

```
class Printer(PoweredDevice):
    tipoHojas = None
    Funciones = None

    def __init__(self, modelo, marca, wf, lan, usb, tipo, fun):
        PoweredDevice.__init__(self, modelo, marca, wf, lan, usb)
        self.tipoHojas = tipo
        self.Funciones = fun

    def __str__(self): ## \n salto de linea --> alt+92 = \
        return f'TipoHojas: {self.tipoHojas} \nFunciones: {self.Funciones} \n'

    def Imprimir(self):
        print("Ha iniciado las impresiones")

Printer1 = Printer('Printeer1', 'PrintpRo', 'SI', 'SI', 'SI', 'Carta-Oficio', 'B/N-Color')
print(Printer1)
Printer1.Imprimir()
```

```
class Copier(PoweredDevice):
    cantiMax = None
    tipoCopia = None

    def __init__(self, modelo, marca, wf, lan, usb, canti, tipoC):
        PoweredDevice.__init__(self, modelo, marca, wf, lan, usb)
        self.cantiMax = canti
        self.tipoCopia = tipoC

    def __str__(self): ## \n salto de linea --> alt+92 = \
        return f'Cantidad Maxima: {self.cantiMax} \nTipoCopia: {self.tipoCopia} \n'
```

```
    def CopierBN(self):
        print("Ha iniciado la copia BN")

    def CopierColor(self):
        print("Ha iniciado la copia color")
```

```
Copier1 = Copier('Scan1','SacnnerpRo', 'NO', 'NO', 'NO', 'Max:1000Hojas', 'B/N-Color')
print(Copier1)
Copier1.CopierBN()
Copier1.CopierColor()
```

```
PoweredDevice1.apagar()
```



Mascota

Dueño

Historial

Parte Práctica.

16. Ejercicio de Planteamiento.

- Identificar un problema cualquiera del mundo real.
- Mostrar el uso de encapsulación.
- Mostrar el uso de la herencia simple.
- Mostrar el uso de herencia múltiple.

Parte Práctica.

16. Ejercicio de Planteamiento.

- Identificar un problema cualquiera del mundo real.
- Mostrar el uso de encapsulación.
- Mostrar el uso de la herencia simple.
- Mostrar el uso de herencia múltiple.

```
class Mascota:
    nombreMascota = None
    edadMascota = int
    tipoMascota = None

    # constructor
    def __init__(self, name, edad, tipo):
        self.nombreMascota = name
        self.edadMascota = edad
        self.tipoMascota = tipo

    def __str__(self): ## \n salto de linea --> alt+92 = \
        return f'NombreMasctoa: {self.nombreMascota} \nEdad: {self.edadMascota} \nTipoMascota:{self.tipoMascota}'
    # encapsulacion
    ## getters
    def get_nombre(self):
        return self.nombreMascota

    ## setters
    def set_nombre(self, nuevo_nombre):
        self.nombreMascota = nuevo_nombre
    ##instanciar

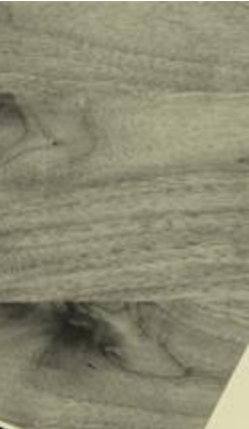
Mascota1 = Mascota('Perro',10,'Canino')
print(Mascota1)
### herencia
class Duenio(Mascota):
    ci_duenio = int
    nombreDuenio = None
    celularDuenio = int

    def __init__(self, nameM, edadM, tipoM, ci, nameD, celD):
        Mascota.__init__(self, nameM, edadM, tipoM)
        self.ci_duenio = ci
        self.nombreDuenio = nameD
        self.celularDuenio = celD


    def __str__(self): ## \n salto de linea --> alt+92 = \
        return f'ciD: {self.ci_duenio} \nNombreDue: {self.nombreDuenio} \n Cel: {self.celularDuenio}\n'

Duenio1 = Duenio('PerroPrueba',12,'Felino',2132,'Antonio',321)
print(Duenio1)
class HistorialClinico(Mascota):
    nroHistorial = int
    doctorHistorial = None
    def __init__(self,nameM, edadM, tipoM, nroH, doctor):
        Mascota.__init__(self, nameM, edadM, tipoM)
        self.nroHistorial = nroH
        self.doctorHistorial = doctor
    def __str__(self):
        return f'NombreMascota: {self.nombreMascota} \nEdad: {self.edadMascota} \nTipoMascota:{self.tipoMascota}\nNroHistorial:{self.nroHistorial}\nDoctor:{self.doctorHistorial}'

HistorialClinico1 = HistorialClinico('Perro',9,'Canino',23,'RicardoPerez')
print(HistorialClinico1)
```



```
NombreMasctoa: Perro
Edad: 10
TipoMascota:Canino
ciD: 2132
NombreDue: Antonio
Cel: 321
```



```
NombreMascota: Perro
Edad: 9
TipoMascota:Canino
NroHistorial:23
Doctor:RicardoPerez
```


A black and white photograph of a person's hands clapping. The person is wearing a plaid shirt. In the foreground, there is a wooden desk with an open notebook and a smartphone resting on it. A tablet is also visible in the background. A dark grey rectangular box with a thin white border is overlaid on the left side of the image, containing the text '¡GRACIAS!'.

¡GRACIAS!