TAREA HITO 2

Integrante: Cristian Josue Poma Zuleta

Docente: William Roddy Barra Paredes

Asignatura: Programación De Sistemas Embebidos

MANEJO DE CONCEPTOS

1. **Qué es un sistema embebido?**

- Un sistema embebido es un sistema de computación basado en un microprocesador o un microcontrolador diseñado para realizar una o algunas pocas funciones dedicadas, frecuentemente en un sistema de computación en tiempo real.

2, ¿Mencione 5 ejemplos de sistemas embebidos?

- Sistemas de calefacción central.
- Sistemas GPS.
- Cajeros automáticos.
- Dispositivos médicos.
- Sistemas de automoción.

3, **¿Menciona las diferencias o similitudes entre un sistema operativo, un sistema móvil y un sistema embebido?**

Sistema Operativo: Es el conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación de software

Sistema móvil: Es un conjunto de programas de bajo nivel que permite la abstracción de las peculiaridades específico del teléfono móvil.

Sistema embebido: Es un sistema electrónico diseñado para realizar pocas funciones en tiempo real, según sea el caso, y se diseñan para cubrir necesidades específicas.

4, **¿A que se referirán los términos MCU y MPU? Explique cada una de ellas.**

MCU: Se refiere a un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales que cumplen una tarea específica.

MPU: Se refiere al chip que se encuentra integrado en la placa base y que se encarga de ejecutar las instrucciones que ordena el usuario, su meta del microprocesador es llevar a cabo las órdenes que se vayan dando por parte del usuario del dispositivo vía sistema operativo.

5, ¿Cuáles son los pilares de POO?

Abstracción: Es cuando separamos los datos de un objeto para luego generar un molde (una clase).

Encapsulamiento: Lo puedes utilizar cuando deseas que ciertos métodos o propiedades sean inviolables o inalterables.

Herencia: Nos permite crear nuevas clases a partir de otras. Si tuviéramos una clase “Autos” y quisiéramos crear unas clases “Auto deportivo” o “Auto clásico”, podríamos tomar varias propiedades y métodos de la clase “Autos”. Esto nos da una jerarquía de padre e hijo.

Polimorfismo: Proviene de Poli = muchas, morfismo = formas. Se utiliza para crear métodos con el mismo nombre, pero con diferente comportamiento.

7, ¿Mencione los componentes en lo que se basa POO?. Y explicar cada una de ellas.

Clases: Las clases pueden ser definidas como un molde que contendrá todas las características y acciones con las cuales podemos construir N cantidad de objetos.

Propiedades: Las propiedades son las características de una clase, tomando como ejemplo la clase humanos, las propiedades podrían ser: nombre, el género, la altura, color de cabello, color de piel, etc.

Métodos: Los métodos son las acciones que una clase puede realizar, siguiendo el mismo ejemplo anterior, estas podrían ser: caminar, comer, dormir, soñar, respirar, nadar, etc.

Objetos: Son aquellos que tienen propiedades y comportamientos, estos pueden ser físicos o conceptuales.

8, Defina los siguientes: ○ Multiplataforma. ○ Multiparadigma. ○ Multiproposito. ○ Lenguaje interpretado

- Multiplataforma.

Se denomina multiplataforma a un atributo conferido a programas informáticos o métodos y conceptos de cómputo que son implementados, y operan internamente en múltiples plataformas informáticas.

- Multiparadigma.

Es una práctica que emerge como resultado de la coexistencia de los paradigmas orientado a objetos, procedural, declarativo y funcional buscando mejorar la producción en el desarrollo de proyectos

- Multipropósito.

Es mayormente utilizado como un gestor de contactos para dispositivos iOS como iPhone

- Lenguaje interpretado

Es un lenguaje de programación para el que la mayoría de sus implementaciones ejecuta las instrucciones directamente, sin una previa compilación del programa a instrucciones en lenguaje máquina.

8, Defina a que se refiere cuando se habla de encapsulación y muestre un ejemplo(Código en Python).

En programación orientada a objetos, se denomina encapsulamiento al ocultamiento del estado, es decir, de los datos miembro, de un objeto de manera que sólo se puede cambiar mediante las operaciones definidas para ese objeto.

Ejemplo codigo en Python:

```
class Prueba:
    __atributo_privado = "Atributo fuera de alcance"

    def __metodo_privado(self):
        print("Método fuera de alcance")

    def atributo_publico(self):
        return self.__atributo_privado

    def metodo_publico(self):
        return self.__metodo_privado()


e = Prueba()
print(e.atributo_publico())
e.metodo_publico()
```

9, Defina a que se refiere cuando se habla de herencia y muestre un ejemplo(Código en Python).

La herencia es el mecanismo por el cual una clase permite heredar las características (atributos y métodos) de otra clase. La herencia permite que se puedan definir nuevas clases basadas de unas ya existentes a fin de reutilizar el código, generando así una jerarquía de clases dentro de una aplicación.

```
class Persona:
    fullname: None
    lastname: None
    age: None

    def __init__(self, fullname, lastname, age):
        self.fullname = fullname
        self.lastname = lastname
        self.age = age

    def __str__(self):
        return f'fullname:{self.fullname}, \nlastname:{self.lastname}, \nage:{self.age} '

    def set_edad(self, nueva_edad):
        self.age = nueva_edad


per1 = Persona('Cristian', 'Poma', 20)
print(per1)
```

Ejemplo codigo en Python:

```
class Estudiante(Persona):

    curso = None
    email = None

    def __init__(self, fullname, lastname, age, curso, email):
        Persona.__init__(self, fullname, lastname, age)
        self.curso = curso
        self.email = email

    def __str__(self):
        return Persona.__str__(self) + f' \nCurso{self.curso} \nCorreo:{self.email}'

    def modifica_edad(self, nueva_edad):
        Persona.set_edad(self, nueva_edad)
```

```
est1 = Estudiante('Cristian', 'Poma', 20, '214- Lab comp 2', 'jos45@gmail.com')
print(est1)
est1.modifica_edad(40)
print(est1)


class Administrativo:

    sueldo = None

    def __init__(self, fullname, lastname, age, sueldo):
        Persona.__init__(self, fullname, lastname, age)
        self.sueldo = sueldo

    def __str__(self):
        return Persona.__str__(self) + f' \nSueldo:{self.sueldo} '


suel1 = Administrativo('Cristian', 'Poma', 20, '100')
print(suel1)
```

10, Defina los siguientes:

- **Que es una Clase**

Una clase es una plantilla para la creación de objetos de datos según un modelo predefinido.

- **Que es un Objeto**

Un objeto es un ente orientado a objetos que consta de un estado y de un comportamiento, que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.

- **Que es una instancia.**

La instancia es todo objeto que derive de algún otro. De esta forma, todos los objetos son instancias de algún otro.

PARTE PRACTICA

11. Llevar el siguiente código JAVA a Python

Codigo en Java:

```java
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);
  }
}
```

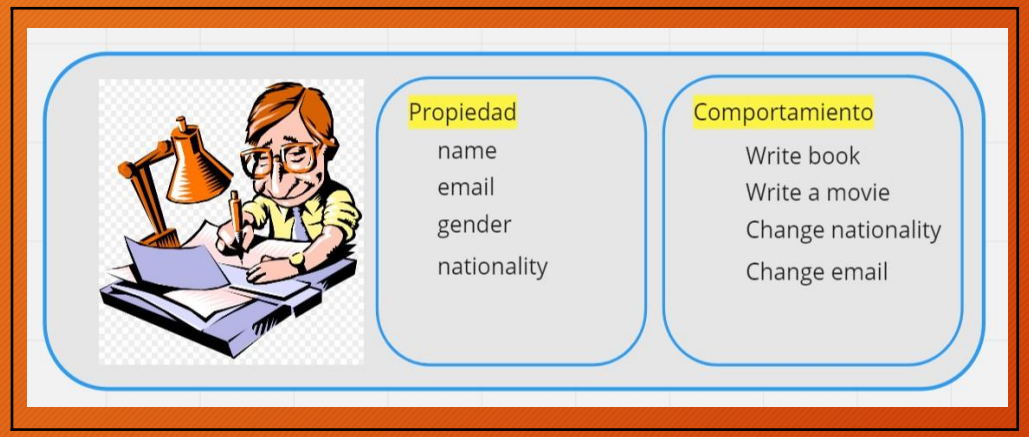
Codigo en Python:

```python
class Main:
    print("Introduce dos numeros")
    first = 10
    second = 20

    print(str(first) + " " + str(second))

    sum = first + second
    print("La suma es: " + str(sum))
```

Funcionamiento:

```
"C:\Users\Josue Poma\PycharmProjects\py
Introduce dos numeros
10 20
La suma es: 30

Process finished with exit code 0
```

12. Crear el código JAVA y Python para el siguiente análisis.



Codigo en Python:

```python
class Libro:
    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

    def Write_Book(self):
        print('Escribe un libro')

    def Write_A_Movie(self):
        print('Escribir una pelicula')

    def Change_Nationality(self):
        print('Cambiar Nacionalidad')

    def Change_Email(self):
        print('Cambiar Email')
```

```python
    def __str__(self):
        return"Name: {} \nEmail: {} \nGender: {} \nNationality: {}".format(self.name, self.email, self.gender,
        self.nationality, self.Write_Book(), self.Write_A_Movie(), self.Change_Nationality(), self.Change_Email())


lec1 = Libro('Cristian', 'jos45@gmail.com', 'Hombre', 'Colombiano')
print(lec1)
Libro.Write_Book
Libro.Write_A_Movie
Libro.Change_Nationality
Libro.Change_Email
```

Funcionamiento:

```
Escribe un libro
Escribir una película
Cambiar Nacionalidad
Cambiar Email
Name: Cristian
Email: jos45@gmail.com
Gender: Hombre
Nationality: Colombiano
```

13. Crear un programa Python que genere los primeros N números de la serie fibonacci.

- El programa tiene que leer un valor por consola.

  Ejem: **N = 8**

- Para el valor leído anteriormente, la salida debería ser:

  0, 1, 1, 2, 3, 5, 8, 13,

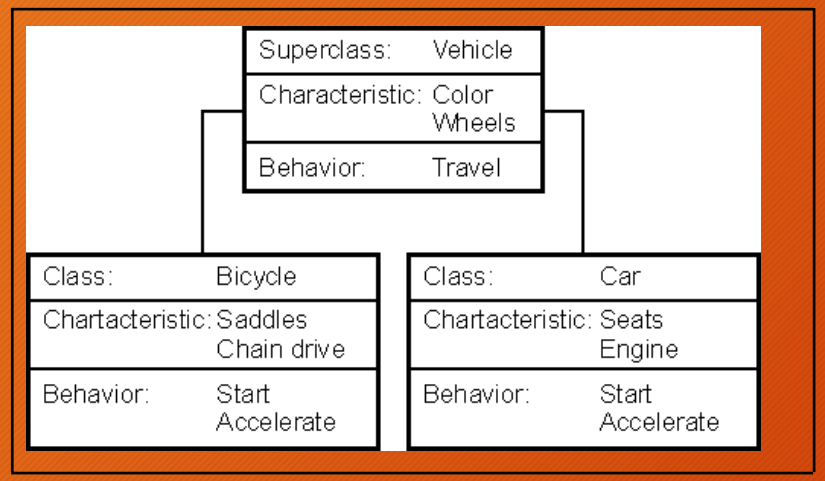
Codigo en Python:

```python
def fibonacci(nu):
    arr = [0, 1]
    if nu == 1:
        print('0')
    elif nu == 2:
        print('[0,', '1]')
    else:
        while (len(arr) < nu):
            arr.append(0)
        if (nu == 0 or nu == 1):
            return 1
        else:
            arr[0] = 0
            arr[1] = 1
            for i in range(2, nu):
                arr[i] = arr[i - 1] + arr[i - 2]
            print(arr)


fibonacci(nu=int(input("Ingrese N: ")))
```

Funcionamiento:

```
Ingrese N: 8
[0, 1, 1, 2, 3, 5, 8, 13]

Process finished with exit code 0
```

14, POO - Crear las clases necesarias para resolver el siguiente planteamiento.




```python
class Vehiculo():
    color = None
    wheels = None

    def __init__(self, color, wheels):
        self.color = color
        self.wheels = wheels

    def __str__(self):
        return "Color Vehiculo: {}, Cantidad llantas  {} ".format( self.color, self.wheels )
```

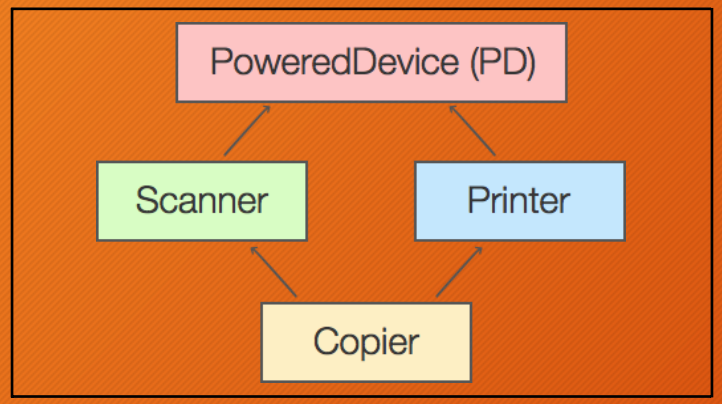
```python
class Car(Vehiculo):

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

    def __start__(self):
        print("Prendiendo vehiculo")

    def __accelerate__(self):
        print("En marcha el vehiculo")

    def __str__(self):
        return Vehiculo.__str__(self) + ", {} milla/dia, {} c".format(self.seats, self.engine, self.__start__(), self.__accelerate__())


c = Car("verde", 4, 300, 500)
print(c)
Car.__start__
Car.__accelerate__
```

```python
class Bicycle(Vehiculo):

    def __init__(self, color, wheels, saddles, chain):
        Vehiculo.__init__(self, color, wheels)
        self.saddles = saddles
        self.chain = chain

    def __startb__(self):
        print("Preperando Bicicleta")

    def __acceleprateb__(self):
        print("Puesto en marcha bicicleta")

    def __str__(self):
        return Vehiculo.__str__(self) + ", {} asientos, {} cond.".format(self.saddles, self.chain, self.__startb__(), self.__acceleprateb__())


b = Bicycle("blanco", 5, 2, 7)
print(b)
Bicycle.__startb__
Bicycle.__acceleprateb__
```

Funcionamiento:

```
Prendiendo vehiculo
En marcha el vehiculo
Color Vehiculo: verde, Cantidad llantas  4 , 300 milla/dia, 500 c
Preperando Bicicleta
Puesto en marcha bicicleta
Color Vehiculo: blanco, Cantidad llantas  5 , 2 asientos, 7 cond.

Process finished with exit code 0
```

15. Realizar un análisis para el siguiente escenario.




```python
class poweredDevice:
    nivel_de_potencia: None
    estandares: None
    control: None

    def __init__(self, nivel_de_potencia, estandares, control):
        self.nivel_de_potencia = nivel_de_potencia
        self.estandares = estandares
        self.Control = control

    def __str__(self):
        return f'Datos\nnivel_de_potencia:{self.nivel_de_potencia},' \
               f' \nestandares:{self.estandares}, \nControl:{self.Control}

    def set_edad(self, nueva_edad):
        self.age = nueva_edad


power1 = poweredDevice('25W', '802.3at Tipo 2', 'PD')
print(power1)
```

```python
class Scanner(poweredDevice):

    tipo_s = None
    calidad = None
    marca = None

    def __init__(self, nivel_de_potencia, estandares, control, tipo_s, calidad, marca):
        poweredDevice.__init__(self, nivel_de_potencia, estandares, control)
        self.tipo_s = tipo_s
        self.calidad = calidad
        self.marca = marca

    def __str__(self):
        return poweredDevice.__str__(self) + f' \nTipo_s:{self.tipo_s} \nCalidad:{self.calidad} \nMarca:{self.marca}'


est1 = Scanner('25W', '802.3at Tipo 2', 'PD', 'Mediano', 'Alta', 'Epson')
print(est1)
```

```python
class Printer(poweredDevice):

    Capacidad_hojas = None
    Modelo = None
    Tipo_P = None

    def __init__(self, nivel_de_potencia, estandares, control, Capacidad_hojas, Modelo, Tipo_P):
        poweredDevice.__init__(self, nivel_de_potencia, estandares, control)
        self.Capacidad_hojas = Capacidad_hojas
        self.Modelo = Modelo
        self.Tipo_P = Tipo_P

    def __str__(self):
        return poweredDevice.__str__(self) + f' \nCapacidad_hojas:{self.Capacidad_hojas} ' \
                                             f'\nModelo:{self.Modelo} \nTipo_P:{self.Tipo_P}'


pri1 = Printer('25W', '802.3at Tipo 2', 'PD', 200, 'epson', 'grande')
print(pri1)
```

```python
class Copier(Scanner, Printer):

    Color_C = None
    Marca_C = None
    Calidad_C = None

    def __init__(self, tipo_s, calidad, marca, Capacidad_hojas, Modelo, Tipo_P, Color_C, Marca_C, Calidad_C):
        Scanner.__init__(self, tipo_s, calidad, marca)
        Printer.__init__(self, Capacidad_hojas, Modelo, Tipo_P)
        self.Color_C = Color_C
        self.Marca_C = Marca_C
        self.Calidad_C = Calidad_C

    def __str__(self):
        return Scanner.__str__(self) + f' \nColor_C:{self.Color_C} \nMarca_C:{self.Marca_C} \nCalidad_C:{self.Calidad_C}'


cop1 = Copier('Mediano', 'Alta', 'Epson', 200, 'epson', 'grande', 'Negro', 'epson', 'Baja')
print(cop1)
```

Funcionamiento:

```
Datos
nivel_de_potencia:25W,
estandares:802.3at Tipo 2,
Control:PD
Datos
nivel_de_potencia:25W,
estandares:802.3at Tipo 2,
Control:PD
Tipo_s:Mediano
Calidad:Alta
Marca:Epson
Datos
nivel_de_potencia:25W,
estandares:802.3at Tipo 2,
Control:PD
Capacidad_hojas:200
Modelo:epson
Tipo_P:grande
```

16. Ejercicio de planteamiento.

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

```
class Usuarios:
    nid: None
    nombres: None
    apellidos: None
    email: None
    password: None

    def __init__(self, nid, nombres,
apellidos, email, password):
        self.nid = nid
        self.nombres = nombres
        self.apellidos = apellidos
        self.email = email
        self.password = password

    def __str__(self):
        return f'__Datos__\nId:{self.nid},
\nNombre:{self.nombres},
\nApellidos:{self.apellidos},
\nEmail:{self.email},
\nPassword:{self.password} '


usu1 = Usuarios('N16', 'Cristian', 'Poma',
'jos@gmail.com', 'io8489yguyg')
print(usu1)
```

```
class Docente(Usuarios):

    asignatura = None
    profecion = None

    def __init__(self, nid, nombres, apellidos, email,
password, asignatura, profecion):
        Usuarios.__init__(self, nid, nombres,
apellidos, email, password)
        self.asignatura = asignatura
        self.profecion = profecion

    def __str__(self):
        return Usuarios.__str__(self) + f'
\nAsignatura:{self.asignatura}
\nProfecion:{self.profecion}'


doc1 = Docente('N16', 'Cristian', 'Poma',
'jos@gmail.com', 'io8489yguyg', 'Fisica', 'Ingeniero')
print(doc1)


class Estudiante(Usuarios):

    carrera = None

    def __init__(self, nid, nombres, apellidos, email,
password, carrera):
        Usuarios.__init__(self, nid, nombres,
apellidos, email, password)
        self.carrera = carrera

    def __str__(self):
        return Usuarios.__str__(self) + f'
\nCarrera:{self.carrera} '


est1 = Estudiante('N16', 'Cristian', 'Poma',
'jos@gmail.com', 'io8489yguyg', 'Ing. Sistemas')
print(est1)
```

```
class Curso(Docente, Estudiante):

    numero = None
    paralelo = None

    def __init__(self, nid, nombres,
apellidos, email, password, asignatura,
profecion, carrera, numero, paralelo):
        Docente.__init__(self, nid,
nombres, apellidos, email, password,
asignatura, profecion)
        Estudiante.__init__(self, nid,
nombres, apellidos, email, password,
carrera)
        self.numero = numero
        self.paralelo = paralelo

    def __str__(self):
        return f'Usuario: {self.nid}
{self.nombres} {self.apellidos}\nEmail:
{self.email}\nPasword:
{self.password}\nAsignatura:
{self.asignatura}\nProfecion:
{self.profecion}\nCarrera:{self.carrera}\nN
umero de
Curso:{self.numero}\nParalelo:{self.paralel
o}'


cur1 = Curso('N16', 'Cristian', 'Poma',
'jos@gmail.com', 'io8489yguyg','Fisica',
'Ingeniero', 'Ing._Sistemas', 245, '2b')
print(cur1)
```

Funcionamiento:

```
Email:jos@gmail.com,
Password:io8489yguyg
__Datos__
Id:N16,
Nombre:Cristian,
Apellidos:Poma,
Email:jos@gmail.com,
Password:io8489yguyg
Asignatura:Fisica
Profecion:Ingeniero
__Datos__
Id:N16,
Nombre:Cristian,
Apellidos:Poma,
Email:jos@gmail.com,
Password:io8489yguyg
Carrera:Ing._Sistemas
Usuario: N16 Cristian Poma
Email: jos@gmail.com
Pasword: io8489yguyg
Asignatura: Fisica
Profecion: Ingeniero
Carrera:Ing._Sistemas
Numero de Curso:245
Paralelo:2b
```

GRACIAS