INGENIERIA EN SISTEMAS

“PRACTICA FINAL HITO 2”

MATERIA	***:***	***“PROGRAMACIONDE SISTEMAS EMBEBIDOS“***
DOCENTE	***:***	***ING. WILIAM BARRA PAREDES***
REALIZADO POR	***:***	***KEVIN ALVARO HUAYLLANI LOPEZ***

LAPAZ - BOLIVIA

2020

TRABAJO PRACTICO HITO 2

1. ¿Qué es 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.
- Rastreadores de fitness.
- Dispositivos medicos.

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

Tanto un sistema operativo móvil y normal tiene que ver con como las compañías tecnológicas individuales han implementado varias versiones de los sistemas operativos que proporcionan los entornos fundamentales para las aplicaciones de software tradicionales, así como las nuevas aplicaciones móviles.

La principal diferencia entre un ordenador tradicional o movil y un sistema embebido está en la optimización del espacio y sus características técnicas.

4. ¿A qué se referirán los términos MCU y MPU?

- **Explicar a que se refiera cada una de ellas.**

Mcu: es el que describe un cuerpo que se mueve alrededor de un eje de giro con un radio y una velocidad angular (ω) constantes, trazando una circunferencia y con una aceleración centrípeta.

Mpu: puede referirse a: Multiple Process Unit, microchip procesador diseñado para realizar tareas múltiples dentro de un sistema de cómputo. Unidad de microprocesador (en inglés microprocessor unit) o unidad central de procesamiento.

Ejemplos:

MCU

El movimiento circular uniforme está presente en multitud de situaciones de la vida cotidiana.

- como el plato del microondas.

- las ruedas de nuestros vehículos entre muchos otros.

MPU:

Mpu 6050: es una pequeña pieza tecnológica de procesamiento de movimiento.

Mpu 60XO: *es la primera solución del mundo en procesamiento de movimientos, con la fusión integrada de sensores con 9-ejes*

5. ¿Cuáles son los pilares Poo?

Son:

ABSTRACCION:

El ejemplo dado captura la esencia del concepto de abstraer. Cuando hacemos una abstracción, queremos omitir detalles que no son necesarios para nosotros, y queremos solamente mostrar lo que sí es relevante.

ENCAPSULAMIENTO:

puedes utilizar clases para modelar entidades las cuales son relevantes para tu aplicación, sabes además que puedes guardar datos dentro de objetos, y también ejecutar funcionalidad.

HERENCIA:

(Compartir código). La herencia es una relación especial entre dos clases, la clase base y la clase derivada, en donde la clase derivada obtiene la habilidad de utilizar ciertas propiedades y funcionalidades de la clase base, incluso pudiendo sustituir funcionalidad de la clase base.

POLIMORFISMO:

En nuestro caso llamamos polimorfismo cuando un método recibe un parámetro que abarca varios tipos.

6. cuales los componentes en lo que se basa POO?

Son :

Objeto: Se trata de un ente abstracto usado en programación que permite separar los diferentes componentes de un programa, simplificando así su elaboración, depuración y posteriores mejoras.

Métodos: Son aquellas funciones que permite efectuar el objeto y que nos rinden algún tipo de servicio durante el transcurso del programa.

Eventos: Son aquellas acciones mediante las cuales el objeto reconoce que se está interactuando con él.

Atributos: Características que aplican al objeto solo en el caso en que él sea visible en pantalla por el usuario; entonces sus atributos son el aspecto que refleja, tanto en color, tamaño, posición, si está o no habilitado.

7. **Defina los siguientes:**

- **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 el cual soporta más de un paradigma de programación. Según lo describe Bjarne Stroustrup, permiten crear "programas usando más de un estilo de programación".

- **Multipropósito:** Existen dispositivos de red que realizan más de una función. Resulta más cómodo adquirir y configurar un dispositivo que satisfaga todas sus necesidades que comprar un dispositivo para cada función.

- **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).**

Decimos que el encapsulamiento en la programación orientada a objetos es cuando limitamos el acceso o damos un acceso restringido de una propiedad a los elementos que necesita un miembro y no a ninguno más.

El elemento más común de encapsulamiento son las clases, donde encapsulamos y englobamos tanto métodos como propiedades.

Ejemplo:

muy común de encapsulamiento son los getters y setters de las propiedades dentro de una clase. Por defecto nos dan el valor "normal" pero podemos modificarlos para que cambie.

private decimal _velocidadActual { get; set; }

public decimal VelocidadActual

```
{
    get{
        return _velocidadActual + 2;
    }
    set{
        _velocidadActual = value;
    }
}
```

En el ejemplo que acabamos de ver tenemos dos propiedades, ambas hacen referencia a la velocidad actual, pero hay ligeras diferencias.

Una es privada, por lo que desde fuera de la clase no podemos acceder a su valor.

La segunda es pública y accede a la privada anteriormente mencionada.

El ejemplo que hemos visto es un caso real, los coches suelen marcar un par de kilómetros por hora mas de los reales. Por lo que encapsulamos esa lógica dentro del setter, el cual esta oculto para el consumidor que vería en su cuentaquilómetros la velocidad con los dos kilómetros por hora extra.

Podemos decir que encapsulamiento es una forma de ocultación de información entre entidades, mostrándose entre ellas solo la información más necesaria.

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

La *herencia* es un mecanismo de la programación orientada a objetos que sirve para crear clases nuevas a partir de clases preexistentes. Se toman (*heredan*) atributos y comportamientos de las clases viejas y se los modifica para modelar una nueva situación.

La clase vieja se llama *clase base* y la que se construye a partir de ella es una *clase derivada*.

Ejemplo:

a partir de una clase Persona (que contenga como atributos identificacion, nombre, apellido) podemos construir la clase AlumnoFIUBA que extiende a Persona y agrega como atributo el padron.

Para indicar el nombre de la clase base, se la pone entre paréntesis a continuación del nombre de la clase (en lugar de la expresión object que poníamos anteriormente; en realidad `object` es el nombre de la clase base genérica).

Definimos `Persona`:

```
class Persona(object):

    "Clase que representa una persona."

    def __init__(self, identificacion, nombre, apellido):

        "Constructor de Persona"

        self.identificacion = identificacion

        self.nombre = nombre

        self.apellido = apellido

    def __str__(self):

        return " %s: %s, %s" %

            (str(self.identificacion), self.apellido, self.nombre)
```

A continuación definimos `AlumnoFIUBA` como derivada de `Persona`, de forma tal que inicialice el nuevo atributo, pero a su vez utilice la inicialización de `Persona` para las atributos de la clase base:

```
class AlumnoFIUBA(Persona):

    "Clase que representa a un alumno de FIUBA."

    def __init__(self, identificacion, nombre, apellido, padron):
```

```
        "Constructor de AlumnoFIUBA"

        # llamamos al constructor de Persona

        Persona.__init__(self, identificacion, nombre, apellido)

        # agregamos el nuevo atributo

        self.padron = padron
```

Probamos la nueva clase:

```
>>> a = AlumnoFIUBA("DNI 35123456", "Damien", "Thorn", "98765")

>>> print a

DNI 35123456: Thorn, Damien
```

Vemos que se heredó el método `__str__` de la clase base. Si queremos, podemos redefinirlo:

```
def __str__(self):

    "Devuelve una cadena representativa del alumno"

    return " %d: %s, %s" %

        (str(self.padron), self.apellido, self.nombre)
```

Volvemos a probar:

```
>>> a = AlumnoFIUBA("DNI 35123456", "Damien", "Thorn", "98765")

>>> print a

98765: Thorn, Damien
```

De una clase base se pueden construir muchas clases derivadas, así como hemos derivado alumnos, podríamos derivar docentes, empleados, clientes, proveedores, o lo que fuera necesario según la aplicación que estemos desarrollando.

10. Defina los siguientes:

Que es una Clase: una clase es una plantilla para la creación de objetos de datos según un modelo predefinido. Las clases se utilizan para representar entidades o conceptos, como los sustantivos en el lenguaje. Cada clase es un modelo que define un conjunto de variables y métodos apropiados para operar con dichos datos.

Que es un Objeto: Se trata de un ente abstracto usado en programación que permite separar los diferentes componentes de un programa, simplificando así su elaboración, depuración y posteriores mejoras.

Que es una instancia: es la particularización, realización específica u ocurrencia de una determinada clase, entidad o prototipo. En los lenguajes de programación orientada a objetos un objeto es una instancia de una clase. Esto es, un miembro de una clase que tiene atributos en lugar de variables.

PARTE PRACTICA:

INGENEIER MIL DISCULPAS ESTOY CON PROBLEMAS DE MAQUINA E POR ESO QUE NO PUDE DESARROLAR LA PARTE PRACTICA. PERO PARA EL HITO 3 LO TENDRE LISTO INSTALADO LOS PROGRAMAS.