**RESUMEN DEL LIBRO LÓGICA DE PROGRAMACIÓN ORIENTADA A OBJETOS**

**link**: https://elibro-net.bdigital.sena.edu.co/es/ereader/senavirtual/70431?page=17

**Definición POO (PROGRAMACIÓN ORIENTADA A OBJETOS)**

Es la programación que simula los objetos del mundo real. Esto se puede realizar siempre y cuando se sepa las características (Variables Miembro) y las acciones (Métodos) que pueda realizar el objeto, por ejemplo, una persona:

Primero debemos elegir un sustantivo que valla de acuerdo a la persona: Persona -> Esta será nuestra clase.

Segundo debemos saber las características que tiene cualquier persona: Nombre, Edad, C.C, Teléfono -> estas serán las variables

Tercero debemos saber las acciones que hace cualquier persona: Respirar (), Comer(), Hablar(), Pensar() -> estos son los métodos. Se les coloca un paréntesis para hacer referencia de que son métodos

Todo esto se puede representar mediante un diagrama de UML.

La programación orientada a objetos lleva unas propiedades básicas que todo lenguaje que trabaje con POO debe seguir y son:

**ABSTRACCION**

La abstracción es la acción de identificar las cualidades y acciones que un objeto puede realizar.

**ENCAPSULAMIENTO Y OCULTAMIENTO DE LAINFORMACIÓN**

El encapsulamiento nos permite guardar la información que tenemos en nuestras variables, para que puedan ser modificadas exactamente en la clase que deseamos sin modificar a las otras clases. Por ejemplo, podemos modificar olas variables de Burro sin alterar las variables de la clase carro.

**ESPECIFICADORES O MODIFICADORES DE ACCESO**

Los modificadores, nos ayudan a restringir el acceso a las variables del método de una clase. Los podemos utilizar dependiendo de la accesibilidad que nosotros le queramos dar a las variables, y a las clases solo las podemos declarar públicas o sin modificador.

Los modificadores que podemos encontrar son:

Público: Podemos hacer que cualquier clase pueda acceder a este elemento, y se logra identificar con el signo +(más)

Privado: Permite que solo los métodos miembros de la clase puedan acceder al elemento y se identifica con el símbolo - (menos)

Protegido: Podemos hacer que las clases derivadas tengan acceso público y restringe el acceso a las demás clases.

Sin modificar: Podemos acceder a este elemento desde cualquier clase siempre y cuando, el elemento esté en la misma ubicación.

hay que tener en cuenta que estas se definen según el problema a resolver. Por ejemplo, no podemos dejar un dato muy personal cómo sería la dirección o el teléfono como público en cambio datos como el nombre no necesitan estar protegida, porque son muy comunes y no requiere de una protección.

**FORMATO DE UNA CLASE**

La estructura de una clase es la siguiente:

MODIFICADOR\_ACCESO CLASE Nombre\_Clase MODIFICADOR\_ACCESO TIPO variable1 MODIFICADOR\_ACCESO TIPO variable2 …

MODIFICADOR\_ACCESO TIPO\_RETORNO metodo1() …

FIN\_ metodo1

MODIFICADOR\_ACCESO TIPO\_RETORNO metodo2() …

FIN\_ metodo2 FIN\_CLASE

**CONSTRUCTORES**

Los constructores son métodos públicos que se ejecutan automáticamente al crear un objeto. Tiene un tipo de acceso, un nombre y un paréntesis

**CREACIÓN DE OBJETOS**

Los objetos son el resultado de la instanciación de la clase. Se desde tener en cuenta que el constructor se ejecuta automáticamente y se le deben enviar los argumentos que necesite el método.

**DEFINICIÓN DEL METODO PRINCIPAL**

Cuando ya hemos realizado las operaciones podemos usarlas en el método principal. Se pueden llamar con un punto (.). Siempre y cuando las variables hayan sido definidas como públicas. Por ejemplo Objeto1.nombre="Jaider"

**Métodos accesores**

Para proteger los datos, además de colocarlos en privado, también podemos crearles accesores para cuando una clase desea ingresar a las variables privadas, le aparezca la sugerencia de crear dos variables miembros de la clase llamada

**Sobrecarga de métodos**

La sobrecarga de datos nos permite tener varios datos con el mismo nombre en una clase, siempre y cuando cambiemos los parámetros de éste.

Al ejecutarse, se puede identificar a cuál se está invocando según la cantidad y el tipo de parámetros

**Herencia**

La herencia es la reutilización de clases que se pueden pasar las variables a otras clases e incluso crear variables propias. La clase que hereda, es llamada subclase y la clase que le pasa las variables es llamada superclase o clase base

**Tipos de herencia**

Hay dos tipos de herencia, la herencia simple y la herencia compuesta. La herencia simple es donde la subclase hereda de una sola clase base. Mientras que la herencia compuesta es donde la subclase hereda de dos clases bases

**PASOS PARA SOLUCIONAR UN PROBLEMA MEDIANTE POO**

Cómo resolver problemas mediante POO

Para poder resolver problemas usando POO, debemos seguir estos 5 pasos:

1. Identificación de datos, acciones y limitaciones: Debemos analizar muy bien el problema e identificar los datos requeridos para la solución del problema

2. Definición de clases: Debemos buscar sustantivos relevantes que vallan de acuerdo al problema y que no sean datos que se requieran para la solución del problema.

3. Definición del método principal: Se debe analizar cuántas instancias de las clases son necesarias.

4. Creación de seudocódigo: debemos escribir los pasos que debe realizar el código. Tiene que estar lo más detallado posible

5. prueba de escritorio: Debemos probar el código para rectificar que dato de salida sea el que requerimos.