**Leyne Sarena Linares Rodriguez. Ficha: 2142329.**

***Lógica de programación orientada a objetos.***

* **Definición de clase y objeto:** Programar bajo el paradigma POO trata sobre simular objetos que existen en el mundo real. Siendo el objeto cualquier cosa con características específicas que lo definen y acciones que este puede realizar. Y la clase la plantilla con la que nos podemos guiar para poder modelar/materializar dicho objeto, el cual toma partes de los métodos y variables para su funcionamiento. Las clases se nombran como sustantivos en singular (Clase cuenta, clase persona, etc.) Estas poseen variables y métodos (Variables nombradas como sustantivos y métodos nombrados como verbos en infinitivo) Se nombran de esa manera para poder ser más fáciles de identificar a la hora de ejecutar dicha clase, variable o método.

De ser necesario o requerido, se puede hacer una representación gráfica, las clases son denominadas ‘*diagrama de clases’* y los objetos son denominados *‘diagrama de objetos*,’ las cuales puede ser modeladas en UML y ayuda a la representación visual del sistema del software a trabajar.

* **Propiedades:** El POO define un conjunto de propiedades básicas que los lenguajes deben cumplir:
  + Abstracción.
  + Encapsulamiento.
  + Ocultamiento de información.
  + Sobrecarga.
  + Polimorfismo.
  + Herencia.
  + Reutilización.

Detallando apropiadamente las propiedades mencionadas con anterioridad, comenzaríamos con:

* + ***Abstracción:*** Una de las características más básicas del POO.Se define como la acción que permite identificar las cualidades y acciones que un objeto puede realizar. Tomándolo de una clase, usando las características que se quieren utilizar de estos.
  + ***Encapsulamiento y ocultamiento de la información:*** Encapsulamiento, es la propiedad que tienen las clases de agrupar las características y acciones relacionadas con una abstracción, bajo una misma unidad de programación. Con esta propiedad, las clases son vistas como cajas negras, donde se conocen sus características y acciones, pero no sus detalles internos.

En cambio, el ocultamiento de información trata sobre proteger los objetos, restringiendo y controlando el acceso en la clase que los define. Lo cual le permite al programador definir cuales variables y métodos podrán ser visibles para otras clases, asegurándose que el estado interno de un objeto no pueda ser alterado en ningún momento por un factor externo.

La capacidad de restringir y controlar el acceso en las clases se logra por medio de unos modificadores de acceso.

* + ***Modificadores de acceso:*** Los modificadores de acceso de las variables y métodos en una clase. Dependiendo de su nivel de ocultamiento, los especificadores pueden ser:
    - ***(+) Publico:*** Puede ser accedido desde cualquier clase. Si es un método, cualquier clase puede invocarlo.
    - ***(-) Privado:*** Solo se puede accederdesde métodos miembros de la clase donde se encuentra definido, o desde otro método en la misma clase.
    - ***(#) Protegido:*** Proporciona acceso público de las clases derivadas(herencia.) y acceso privado para el resto de clases.
    - ***() Sin modificador:*** Se puede acceder al elemento desde cualquier clase que este en la misma ubicación(carpeta) donde se define la clase.

Las clases no pueden declararse, ni ser protegidas o privadas. Solo se pueden declarar públicas o sin modificador. Las variables y métodos, por el contrario, pueden declararse con cualquiera de los modificadores.

* + ***Formato de una clase:***

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

Para poder definir con mejor claridad el formato básico presentado, necesitamos de un constructor.

* + ***Constructores y destructores:*** El constructor es un método público que tiene el mismo nombre de la clase, y se ejecuta automáticamente al crear un objeto de la clase. Los constructores no tienen valor de retorno ni parámetros de envió. Este se emplea para inicializar las variables o parte de ellas. Los valores utilizados para ello pueden ingresar como parámetros al método constructor o pueden leerse dentro del método. También se puede crear un método constructor sin parámetros de entrada.

Mientras el destructor se ejecute automáticamente al iniciar una clase, existe otro método llamado destructor, se ejecuta automáticamente al destruirse la clase. Este método es opcional, y se utiliza para instrucciones finales, como liberar memoria, cerrar archivos, entre otros. El nombre de este método se forma poniendo una virgulilla “~” al nombre de la clase.

* + ***Creación de objetos:*** Para crear un objeto, se debe instanciar una clase, se debe tener en cuenta que el constructor se ejecuta automáticamente, así que se deben enviar los argumentos necesarios atreves del método.
  + ***Definición del método principal:*** Una vez se han definido las clases necesarias para solucionar un problema, se puede proceder a utilizarlas en el método principal. Usualmente, en este método se realiza el proceso de instanciación y es donde se pueden utilizar las variables y los métodos de las clases. Dentro de este, es posible cambiar o imprimir el valor de las variables de los objetos creados utilizando el operador punto (.) siempre y cuando las variables hayan sido declaradas como públicas.