**INTRODUCCION AL PARADIGMA DE OBJETOS.**

**MODULO I**

**Paradigma**:

Teoria o conjuntos de teorias que sirven de modelo a seguir para resolver problemas.

La POO define los programas por medio de “**clases de objetos**”

**Los objetos combinan:**

- Identidad (nombre de la clase)

- Estado (caracteristicas)

- Comportamiento ( acciones )

Los objetos conviven en un programa y se comunican a traves de mensajes.

Ej de lenguajes de POO

- Java

- PHP

- Phyton

- C#

**Ventajas de la PPO**

- Fomenta la reutilizacion del codigo

- Crear sistemas complejos

- Semejanza al mundo real

- Agiliza el desarrollo de sofware

- Facilita el trabajo en equipo.

**Conceptos basicos de la PPO**

**Objeto**: Entidad provista de un conjunto de atributos (datos) y comportamiento (metodos)

**Clase:** Es el modelo en que se definen la propiedades y comportamiento de un tipo de objeto. (molde) Se usa para ir creando objetos. Esta compuesta por atributos (datos), tambien llamadas variables de instancia o variables miembro que nos indican en que estado se encuentra el objeto. Tambien se compone de metodos (funcones o procedimientos) que indican cual es el comportamiento.

Los atributos de la clase son privados al objeto y su acceso se realiza mediante los metodos definidos para manejar esos atributos.

**Metodo (operacion):** Son acciones contenidas por el objeto, que se asignan en la clase. Determina el comportamiento que puede tener un objeto, el cual fue referido en la clase. Es un algoritmo asociado a un objeto o una clase de objetos que se ejecuta tras la recepcion de un mensaje. Son generalmente verbos. Pueden admitir variables como parametros de entrada y retornar valores de salida o retorno. Los metodos u operaciones modifican el estado de los objetos.

Valores de **entrada** (**datos**): Son valores enviados a un metodo para realizar la operación.

Valores de **salida** o retorno (**informacion**): Es el valor devuelto por un operación luego de realizar un procesamiento. Todos los valores de salida deben tener un tipo de dato asociado. Solo devuelven 1 solo objeto.

**Mensaje:** Es la comunicación hacia un objeto para que este ejecute uno de sus metodos para obtener un resultado.

**Propiedad o atributo:** Es un tipo de dato que define a nuestro objeto. Son las caracteristicas y normalmente cuentan con un dato asociado. Siempre seran sustantivos que haran referencia a nombres, tamaños formas y estados. Las clases definen los atributos y los objetos los completan !

**Estado interno:** Segun sus valores en un momento dado representan el estado de ese objeto en ese momento. Ese estado puede ser modificado a traves de un metodo definido en la misma clase a cual pertenece el objeto.

**Componentes de un objeto:**

- Identidad

- Atributos

- Metodos

- Relaciones

**Representacion de un objeto:**

Va a ser representado por medio de una entidad compuesta por sus atributos y funciones.

**Comunicacion entre objetos.**

Emisor le pide algo al receptor y puede que obtenga una respuesta o no.

**Caracteristicas de la PPO**

- **Abstraccion** (nos permite seleccionar ciertas caracteristicas del objeto atributos o comportamientos según necesitemos)

- **Encapsulamiento** Nos permite reunir todos los elementos que pertenecen a una entidad particular.

- **Herencia** Las clases se relacionan entre si y forman una jerarquia de clasificacion. Los objetos heredan la propiedades y comportamiento de todas las clases a las que pertenecen.

- **Polimorfismo** Se pueden definir clases diferentes en la misma jerarquia de herencias que tienen metodos denominados de forma identica pero se comportan de manera distinta.

**MODELOS**

Los Modelados previos de los sistemas de informacion sirven de intermedio para poder entender el problema y la solucion en la orientacion a objetos

**Diagrama de clases (el mas importante)**

1- Identificar clases y objetos del dominio del problema.

2- Definir el comportamiento de los objetos.

3- Asociar estaticamente los objetos.

4- Definir la jerarquia de herencia de clases

**DIAGRAMA UML** es una herramienta para diagramas de modelado que incluye casos de usos, actividades, iteracion, estados e implementacion. (*drawio*: sofware libre para diagramar)

**MODULO II**

**RELACOINES ESTRUCTURALES**

Son relaciones que se dan entre los objetos de diferentes clases.

**Asociacion** Es una relacion estructural que especifica que los objetos de un elemento estan conectados con los objetos de otro elemento. Dada una asociacion entre dos clases se pude navegar de un objeto de una a un objeto de otra.

**Multiplicidad** de una asociacion determina cuantos objetos de cada tipo intervienen en la relacion. Es el numero de instancias de una clase que se relaciona con una instancia de otra clase. Cada asociacion tiene **dos multiplicadores** (uno en cada extremo).

Ej 1 ------> 1,N

**Relacion Simple** 1 – 1 (uno a uno)

**Relaciones multiples** 1 – N (uno a muchos)

**SOBRECARGA DE OPERACIONES**

Es la aparicion de metodos dentro de una misma clase que se llaman igual, pero realizan acciones diferentes.

Nos permite utilizar el mismo nombre para dos operaciones distintas, que aceptan distinta cantidad y/o tipos de parametros y tienen distinta funcionalidad. El tipo de devolucion no puede ser el mismo. El compilador distingue estas operaciones según el tipo y cantidad de parametros.

Ej: class automovis {

void acelerar () { velocidad += 10; }

void acelerar (int km) { velocidad += km; }

void acelerar (int km, boolean nitro); {

if (nitro) { acelerar(km\*2); }

else { acelerar(km); }

}

**VISIBILIDAD**

Determina el alcance que tiene una clase. Es desde donde puedo acceder a una clase, atributo u operación.

El alcance es fundamental en la POO, ya que permite definir el encapsulamiento de la informacion y el comportamiento. Los metodos, atributos y clases deben tener un solo alcance, tambien llamado **tipo de acceso.**

**Modificadores de visibilidad**

**- defaul (nivel paquete):** Acceso por defecto. Si un elemento clase, metodo o atributo tiene acceso por defecto, solo las clases de su mismo paquete tendran acceso a ellos.

**- public:** Si un atributo, metodo o clase tiene visivilidad publica, es accedido desde cualquier clase. Tener en cuenta que solo podremos ver lo que esta marcado como publico, si una clase es publica y sus atributos no, solo podremos ver la clase sola. Habria que marcar como publicos los atributos que querramos.

**- private:** Solo es aplicable a metodos y atributos. Solo pordra ser accedido por los metodos definidos en la misma clase. Cuando declaramos todos los atributos de un objeto como privado lo estamos **encapsulando.** Para acceder a atributos privados podemos definir las **operaciones** internas del objeto como **publicas** para acceder entonces a los atributos.

**- protected:** Se utiliza en la herencia y es aplicable a atributos y a operaciones. Un atributo o metodo protegido puede ser accedido por cualquier otra clase de su mismo paquete y **ademas** por cualquier **sub clase** de ella independientemente del paquete en que se ecuentre. Las subclases pueden acceder a los miembros protegidos a traves de la **herencia**. Basicamente los metodos protected de una clase padre son accedidos por metodos de una clase hija desde cualquier paquete.

**ENCAPSULAMIENTO**

Permite que todo lo referente a un objeto quede aislado dentro de este y solo se pueda acceder a ello a traves de sus metodos.

Los metodos para exponer el objeto al exterior llevan modificador publico y los atributos privado.

Beneficios:

- **Proteccion de datos sensibles**. Los atributos del obj no pueden ser modificados por otras clases u operaciones externas.

- **Facilidad y flexibilidad en el mantenimiento** de apps. Solo tenemos que modificar los metodos, asi no aleramos a otros objetos relacionados.

**GETTERS Y SETTERS** (Minutuo 19 clase 3)

- **setter:** Es un “metodo” que no retorna nada y posee como argumento cuyo tipo de dato es igual al del atributo que trata. Lo que hace es asignar el valor del parametro al atributo correspondiente.

- **getter:** No posee atributo alguno pero devuelve un dato. Lo que hace es resolver el valor del atributo correspondiente.

En la misma clase se define un get y un set por cada atributo con igual tipo de dato. El ser recibe un parametro y el get recibe un valor. Basicamente, con los setters accedemos al atributo del objeto enviando un parametro, para modificarlo/inicializarlo. Y con el getter los usamos en los metodos para acceder al valor del atributo establecido con el setter.

En UML tenemos,

Clase

atributos

metodos

setter

getter

**CONSTRUCTORES**

- Los objetos son construidos a partir de una clase (molde). Todo objeto depende de una clase.

- Para construir un objeto se necesita un constructor. El constructor crea un objeto en memoria y se le asigna a la variable con el nombre que elijamos.

- Un constructor es el metodo invocado en el momento en que se crean las instancias de una clase, o sea los objetos.

- **Tiene el mismo nombre que la clase** y para invocarlo se utiliza la palabra reservada **new.**

**- No puede tener ningun tipo de devolucion.** Ni siquiera void.

**-** Se pueden sobrecargar, o sea, una clase puede terner mas de un constructor y por lo tanto distintas formas de inicializar los atributos. Distinta cantidad de parametros para que sean validos, o distinto tipo de dato o distinto orden.

- Toda clase debe tener al menos 1 constructor. Si no lo asignamos el compilador lo establecera uno vacio por defecto.

- En sobrecarga de constructores, podemos tener un constructor vacio para ir asignandole valore a los atributos a la vez que los vamos cargando, y a la par podemos tener uno o varios constructores sobrecargados con parametros ya asignados previamente.

- Los constructores parametrizados evitan el uso de inicializacion con setters y getters. Y nos da la seguridad de tener que ingresar todos los parametros que requiere o arrojara error.

**DESTRUCTORES**

sigue clase 3 tiempo 1:23:00